AFRL-RI-RS-TR-2013-097

QUALITY OF INFORMATION ASSURANCE – ASSESSMENT, MANAGEMENT AND USE (QIAAMU)

RAYTHEON BBN TECHNOLGIES

APRIL 2013

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ AIR FORCE MATERIEL COMMAND ■ UNITED STATES AIR FORCE ■ ROME, NY 13441

NOTICE AND SIGNATURE PAGE

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2013-097   HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /

PATRICK M. HURLEY
Work Unit Manager

/ S /

WARREN H. DEBANY, JR.
Technical Advisor
Information Exploitation and
Operations Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* APRIL 2013	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* MAY 2008 – DEC 2012

4. TITLE AND SUBTITLE

QUALITY OF INFORMATION ASSURANCE – ASSESSMENT, MANAGEMENT AND USE (QIAAMU)

5a. CONTRACT NUMBER FA8750-08-C-0196

5b. GRANT NUMBER N/A

5c. PROGRAM ELEMENT NUMBER 62788F & 63788F

5d. PROJECT NUMBER 459G

5e. TASK NUMBER PH

5f. WORK UNIT NUMBER 02

6. AUTHOR(S)

Raytheon BBN Technologies:
Partha Pal, Hala Mostafa, Nathaniel Soule, Michael Atighetchi
Nicholas Hoff, Emily Hahn, Amy Fedyk, Paul Rubel, Kathryn McGuire, Mathew TanCreti

University of Illinois:
Robin Berthier, William Sanders, Nathan Dautenhahn

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Prime:	Sub:
Raytheon BBN Technologies	University of Illinois
10 Moulton Street	1308 W. Main Street
Cambridge, MA 02138	Champaign, IL 61801

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIGA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2013-097

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2013-1558
Date Cleared: 29 March 2013

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project aims to demonstrate the feasibility of run-time mission-oriented assessment of assurance and tradeoff in a distributed system. Distributed mission support systems today incorporate various QoS and security mechanisms, but there is no clear way to judge whether the system is delivering QoS and security assurance at the level needed for mission success. Security is treated as an "all or nothing" quality--which leads to missed opportunities as well as undue risk taking. Measuring security is difficult. QIAAMU took an approach similar to QoS management where assurance is measured against requirements specified by mission stakeholders. This approach enables us to treat QoS and security in a uniform way as well as consider QoS-Security tradeoffs when the system fails to deliver QoS and Security at the required level. Starting with the metrics to observe and measure, this project developed a foundation for run time assessment and tradeoff including an assessment scheme, an innovative distributed tradeoff algorithm and usability support, and instantiated the assessment and tradeoff framework within a security architecture, which underwent a final demonstration and evaluation phase in December 2012. This report is the final technical report for the QIAAMU project and describes the research, development, and evaluation results from the project.

15. SUBJECT TERMS

Mission-oriented assessment, Information assurance, QoS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**PATRICK M. HURLEY**
U	U	U	SAR	165	**19b. TELEPONE NUMBER (*Include area code*)** **(315) 330-3624**

Table of Contents

Section **Page**

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

Identification

The Quality of Information Assurance- Assessment, Management, and Use (QIAAMU) project, funded by the Air Force Research Laboratory (AFRL) under Contract No. **FA8750-08-C-0196** is performed by BBN Technologies (prime contractor) in association with the University of Illinois Urbana-Champaign.

The project started on May 5th 2008, and ended on 31st December 2012.

This document is the Final Report of the project, CDRL No. A007. This report complements the interim final technical report (CDRL No. A008) delivered on April 2011.

1.0 SUMMARY

The *Quality of Information Assurance- Assessment, Management and Use (QIAAMU)* project was an Air Force Research Laboratory (AFRL) funded project that ran from May 5th 2008 to 31st December 2012. BBN Technologies was the prime contractor, with the University of Illinois as a subcontractor.

1.1 Goals of the QIAAMU Project

This project aimed to demonstrate the feasibility of a continuous and mission-oriented assessment of how assured the operations of an information system are, and also the feasibility of combining assurance of service delivery (i.e., managing QoS) and Information Assurance (i.e., managing security), leading to meaningful QoS-security tradeoffs that maximize mission success. Toward that end, we developed a methodology and supporting taxonomy to identify and collect measurements and observations from within the system and its operating environment to use for continuous and mission-oriented assessment, showed that it is feasible to collect the required observations and measurements, aggregate and process them to derive a relativistic assessment indicator in a proof of concept, designed and implemented a distributed tradeoff algorithm to manage the QoS-security tradeoff in a distributed system setting, developed a distributed test bed where the assessment and tradeoff mechanisms were deployed alongside the security mechanisms and resource management mechanisms for measurements and actuation for experimentation and demonstration, and demonstrated and evaluated the utility of the QoS-security tradeoff management in the context of representative mission scenarios running in the test bed.

1.2 Summary of Major Results

The QIAAMU project resulted in significant advancement in the area of information assurance, specifically runtime assessment of information assurance, and synergistic management of the QoS and security. The results can be summarized under the three broad categories as follows:

Research and Development (R&D)

We developed and formalized the concept of a requirements-based mission-oriented runtime assessment that combines both QoS (i.e., service delivery) and security (i.e., information assurance). As part of the requirements-based assessment approach, we developed a methodology and structure for capturing security requirements that is well aligned with the way QoS requirements are represented. The methodology to represent a mission's IA requirements in terms of security attributes (e.g., Confidentiality, Integrity and Availability), the spatial scope the attribute applies to (e.g., Confidentiality of a specific end to end interaction), and the mission participant and mission epoch the requirement is applicable, and measuring the IA delivered by the system against these requirements is a significant departure from the traditional approaches to measure and evaluate security

To support runtime assessment, we developed a taxonomy of measurable and observables that can be used for assessing the delivered levels of QoS and security.

We developed an extensible security assessment framework that is primarily based on commonly available measurements and observations such as the up or down status of security mechanisms and current configuration of security mechanisms that can be configured to offer multiple security postures (i.e., lenient vs. strict firewall, key length etc.). We also showed how other classes of metrics (e.g., logs from security mechanisms) can be used in runtime assessment.

We formulated the QoS-security tradeoff in the context of a distributed system as a distributed constraint optimization problem (DCOP). Existing DCOP solutions make certain assumptions about the dependency structure of the controllable aspects that are not valid for QoS and IA in a distributed system. We developed an enhancement to an existing DCOP solution (called max-sum) to address these challenges.

Experimentation

We performed evaluation experiments periodically throughout the project to track progress and facilitate early course correction. In the initial phase, we experimented with the assessment and a very early preference-based tradeoff in individual blackboards that ignored the global outcome in an emulated network.

We experimented with the tradeoff algorithm in the abstract (i.e., Java implementation of the algorithm, running on a single host without integrating with a distributed system) with synthetic data to study and analyze its performance.

We also performed similar standalone experiments to study the feasibility of using and interoperating with the emerging Security Control and Automation Protocol (SCAP) compliant tools and services, specifically tools and services based on the Open Vulnerability Assessment Language (OVAL) standard.

Finally, we have performed experiments on a distributed test bed implemented using Virtual Box VMs and PfSense routers running close air support scenarios that were developed in house based on published reports. Mission participants use a Publish-Subscribe information management system (IMS) to interact to execute the mission. Failures and effects representing cyber attacks, physical destruction and maintenance scenarios were injected during various phases of the ongoing mission to see how QIAAMU runtime assessment and tradeoff mechanism can adapt the system and mission operation to maximize the overall QoS and IA satisfaction of all the stakeholders. Without QIAAMU assessment and tradeoff driven adaptation, many such attack effects would result in aborting the mission.

Documentation and Publication

As part of the QIAAMU project, we wrote several technical documents, including technical reports (Appendix B) and papers published in journals, conferences, and workshops (Appendix C).

2.0 INTRODUCTION

2.1 Project Objective

Information systems should ideally deliver the desired levels of service and information assurance at all times. However, these are often conflicting goals- the applications delivering service to mission users and the security mechanisms responsible for information assurance share the same set of system resources. In many operational contexts, systems cannot be configured to deliver both high levels of service and information assurance. At the same time, information systems supporting DoD missions operate in contested environments where resources may be limited, damages caused by kinetic attacks may render certain resources less usable or completely unusable, and perhaps more importantly cyber attacks and adversarial behavior may introduce failures, resource outage and risks of exfiltration and corruption.

It is no doubt very important to know whether the system is delivering the level of service and IA required by the mission during mission execution. Degradation of the required or expected level of QoS is often visible to human users who can react and adapt to the changing reality, peer applications may not be able to do so. But the situation is completely different for degradation in security. Existing approaches to measure and evaluate security assurance of an information system primarily focus on organizational maturity, development methodology, and various forms of penetration and stress testing. They are largely subjective and typically done separately i.e., not online while a mission is ongoing. The overall score for the system obtained from such an evaluation process is somewhat meaningless for mission stakeholders who are interested in knowing whether the system is operating at the expected level of assurance during the mission.

Knowing whether the system is delivering the expected level of service and assurance is not enough. System users and operators need to make runtime decisions when they become aware that one or both of IA and QoS requirements are not being met. When security is threatened, the typical response often is to lock down the system and abort the mission. This results in lost opportunity—perhaps some mission operations can continue at the required level of QoS and security, but currently there is no technology support for knowing and managing such situations. When service delivery is threatened, and the mission needs that service, often users and system managers often trade off security to gain QoS. This results in undue risk taking without knowing the consequence of such tradeoff to the information system, to the mission and to other users of the system. Here also, current technology does not offer any help.

The goal of the QIAAMU project was to address this technology gap by designing, developing, demonstrating and evaluating mechanisms, techniques and algorithms to

- Assess whether a mission-support information system is delivering the level of service and assurance required for the mission at runtime—i.e., when the mission is being executed, and
- Perform tradeoff between assurance and service quality based on preferences applicable for the mission when only some of the requirements can be met simultaneously.

QIAAMU technology combined with the survivability architecture of the mission support information system leads to more assured mission operation, lesser need for mission stakeholders to rely on risky guesswork about the systems current quality of service and assurance state, and better utilization of the system's resources and security mechanisms.

Subjective assessment is an inescapable reality of assessing something like assurance or security that is inherently related to users' perception. But our contribution is that we decompose the overall IA assessment problem into evaluating observable and directly measurable metrics against specified requirements similar to what is commonly done with QoS. One can argue that our decomposition of IA and QoS attributes and causal structures are also subjective, but the assessment is not. We claim that if the security engineer consistently applies the decomposition and uses the same causal structure, our approach leads to effective and internally consistent assessment and meaningful tradeoffs between information assurance and the quality of service.

2.2 Project Overview

Raytheon BBN was the prime contract, supported by the University of Illinois as a subcontractor. The project was executed over a 4.5 year period (May 5th 2008 to 31st December 2012). We developed the conceptual framework of a runtime assessment that compares relevant measurements and observations from the system at runtime to the IA and QoS requirements specified by the mission users. We developed the QoS-IA tradeoff problem as a DCOP, and developed a distributed message passing extension to a well known algorithm to solve it. We prototyped the assessment and the distributed tradeoff mechanisms, and integrated them with the survivability architecture of an existing distributed system emulating an in house close air support inspired mission. The resulting system is used to evaluate the QIAAMU runtime assessment and tradeoff management by injecting faults and attack effects during various phases of the ongoing mission. We also focused on a specific metric class—namely the reports produced by security mechanisms, specifically authentication mechanisms, with the objective to continuously assess the likelihood of account compromises as the users continue to use the information services protected by the authentication mechanism. The resulting account monitoring system is being transitioned to University of Illinois internal network to monitor thousands of users including students and faculty. We published research findings and results in technical papers and participated in transition discussions. We also identified shortcomings and developed a plan for the next steps.

2.3 Report organization

The rest of this document is organized as follows. Section 3 documents the R&D methods, solution approaches and assumptions. Section 4, organized in several detailed technical subsections delves into the research findings and evaluation results. Section 5 concludes the report.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

This section describes our research methods, assumptions, and procedures. This includes the scope of our research and organization of the research project, as well as the methods and procedures we undertook to conduct the research, develop our prototype software, and evaluate the QIAAMU software prototype.

3.1 The Need for Runtime Assessment

Advanced mission-critical systems incorporate multiple security mechanisms in order to maintain mission-specific security requirements (often described in terms of levels of availability, integrity or confidentiality), organized in a coherent manner that facilitates defense in depth. However, quantifying the contribution of the included security mechanisms (e.g., firewalls, antivirus scanners, access control, encryption etc.) or the underlying survivability architecture in improving the system's ability to thwart, defend or survive cyber attacks has eluded researchers and practitioners for a long time. As a result, *information assurance* (IA)—assurance that the security mechanisms are effective, and the system can be entrusted with critical information processing tasks—is largely *qualitative*, and is estimated primarily by *offline* analyses, testing, modeling and experimentation that are completely detached from the deployed system and the missions it supports. Consequently, at runtime or during mission execution, arguably the time when it is most critical to be assured about the system, IA takes on an *all-or-nothing* flavor (i.e., either the entire system is assured or it is not) and is largely dependent on the user's *perception* (as opposed to any real and objective measure). The risks posed by attack-induced failures and security-compromises, environmental threats such as the release of a new virus or discovery of a new vulnerability, and user-made decisions to change security settings or to bypass security entirely are neither well understood nor considered.

DoD systems are constantly under the threat of malicious attacks, and therefore need to incorporate an appropriate collection of defense mechanisms for their security. At the same time, there has been an increasing need to know (estimate) the level of assurance accorded by the defense, especially during mission operation. Without such knowledge and given the *perception-based* and *all-or-nothing* view of IA, war fighters and system owners might:

- Incorrectly decide not to use the information system, because they fear a compromise,
- Change the system configuration improperly without understanding the impact on the mission,
- Continue to use the system without knowing the extent or significance of a compromise, or
- Think the system is protected (and not act on reported events) when it is not.

All of these cases incur risks. With the increasing dependence on network centric information systems, these risks cannot be left unaddressed. The capability to assess whether the QoS and IA requirements of the mission are being met at runtime is therefore very important to the success of network centric war fighting missions.

3.2 Taking Advantage of the Added Awareness Resulting from Assessment

The assurance and QoS requirements of a mission may conflict with each other in various ways. Requirements of multiple mission users may conflict; assurance and QoS requirements of even a single user may conflict with each other. It may be possible to detect (and resolve) some conflicts by examining the high level requirements (e.g., multiple users may be interested in the same physical part of the system and may have conflicting requirement about the same QoS or IA attribute), but not all. Some conflicts will require a deeper analysis of the requirements, for example, a user may need both confidentiality and fast completion time, but the time needed to encrypt and decrypt will make the delay unacceptable, even under the most well provisioned configuration possible in the deployment environment. It is easy to see the limitations of such static analysis, for example conflicts not having enough CPU resource to allocate the tasks from two stakeholders and also the defense mechanisms cannot be detected until runtime. But more importantly, mission workloads can be dynamic, and failures and cyber attacks can change the operating environment at arbitrary times in unpredictable ways—static analysis can only help planning and provisioning for these situations. But there are limits to how much over-provisioning is feasible and novel attacks can throw even the most well planned missions off balance. However, if there is a conflict, it will definitely manifest at runtime, and therefore, runtime management provides the best coverage for handling such conflicts.

The added awareness resulting from QIAAMU runtime assessment is one necessary component for the overall runtime response in the sense that it determines when QoS and assurance conflicts occur at runtime. But additional capabilities such as the ability to affect changes and polices governing such changes are also needed. Runtime response involves changing the system and its operating environment or changing the mission behavior in some way. Therefore, controllable elements (of the system as well as mission operations) need to be identified and interfaced with the assessment mechanism so that changes can be effected when conflicts are detected. Then, such changes need to be controlled and guided, because uncoordinated changes triggered by every detected conflict may themselves conflict with each other (one change undoing another) or with the overall goal of maximizing the likelihood of mission success.

3.3 Tradeoff

When a conflict is detected, i.e., it is determined that two or more requirements are not being met at a given point within the mission, there are basically two possibilities. There is a configuration that meets all requirements, or there is no configuration where all requirements can be met. Instead of halting the mission at that point, which is often the de facto response today, QIAAMU runtime assessment and tradeoff mechanism attempts to guide the mission to a configuration that offers maximal satisfaction to all QoS and security requirements. If it finds a configuration that satisfies all applicable requirements then the mission continues effectively tolerating the attack or failure that caused the conflicting situation. If it finds a configuration that does not satisfy all requirements, but the overall level of satisfaction is better than the original configuration, then some mission operations can still proceed with confidence. However note that the tradeoff analysis may not find a better configuration, in which case the mission's likelihood of success does not change from what it was, and it is up to the mission users whether to abort or continue on risk.

The formalism that QIAAMU uses to guide the runtime response is a distributed tradeoff computation. Independent agents in the system make local actuation decision, cooperating among themselves via message passing, to arrive at the configuration setting for all the actuators in the system that maximizes the satisfaction of all the applicable assurance and QoS requirements. The QoS and assurance requirements and the impact of a configuration setting on measurable security and QoS attributes are treated as constraints. Finding the value assignment for all configurable actuators that minimizes the penalty (or maximizes the level of satisfaction) is treated as a distributed constraint optimization problem (DCOP). We have extended an existing DCOP solution to suite our distributed formulation.

3.4 Ingredients of Our Solution Approach

The end to end solution package of runtime assessment and tradeoff driven actuation requires interaction among a number of key components. First there needs to be a structured way to capture and represent QoS and security requirements of mission users. High level statements like "highest level of security" or "highest level of QoS" is not actionable. One needs to break the requirements based on security or QoS attributes, the system or mission aspects to which these attributes are attached, what relative level of that attribute they can tolerate and whether such requirements vary over time during the mission or not. Second, the assessment against specified requirements must be based on well-defined metrics—measurements and observations that can be obtained from the system and its operating environment. One difficulty is that there is no universally accepted list of metrics that will be applicable to all systems and acceptable to all missions. Therefore, we adopted the approach of formulating a set of metrics classes and defined assessment in terms of the (metric) classes whose metrics (instances) are readily available in most systems. Third, the tradeoff analysis needs to have an internal model of how the impact of setting an actuator to a configuration propagates in the system, affecting the security and QoS of various services and interactions, and interfering with the impact of other actuators. The analysis also needs some information about the relative importance and preference of various requirements to guide the tradeoff. Finally, as we noted already, in order to effect the tradeoff decisions the runtime mechanism will need to interface with controllable elements of the system.

3.5 Architecture

The runtime assessment and tradeoff driven adaptation mechanism, by necessity, has to be integrated with the distributed system that it is monitoring and managing. As shown in Figure 1,

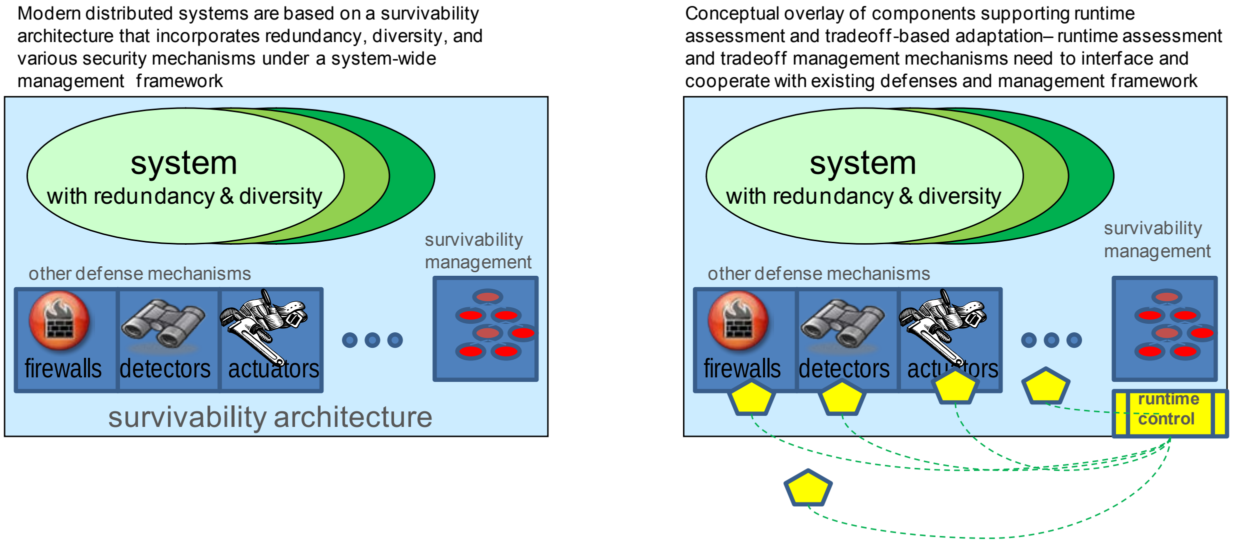

Modern distributed systems are based on a survivability architecture that incorporates redundancy, diversity, and various security mechanisms under a system-wide management framework
system
with redundancy & diversity
other defense mechanisms
survivability management
firewalls
detectors
actuators
survivability architecture
Conceptual overlay of components supporting runtime assessment and tradeoff-based adaptation– runtime assessment and tradeoff management mechanisms need to interface and cooperate with existing defenses and management framework
system
with redundancy & diversity
other defense mechanisms
survivability management
firewalls
detectors
actuators
runtime control

Figure 1: Augmenting survivability architecture with runtime assessment and tradeoff-driven adaptation capability

most modern distributed systems include redundancy, diversity and a number of defense mechanisms under a common management framework. The defense mechanisms perform various functions ranging from network firewalling, file system integrity checking to enforcing security policies including access control at various layers of the system. But, broadly speaking, some of these defense mechanisms are detectors (e.g., intrusion detection mechanisms); others are actuators that respond to external commands (e.g., a firewall accepting an additional rule to block specific IP address) or internal detection events (e.g., policy enforcement mechanisms like SELinux preventing offending interactions upon detection). Existing defenses and system management mechanisms collect a lot of useful measurements and observations that are relevant for runtime assessment—the main missing piece is connecting the understanding of the system's delivered level of IA and QoS to mission requirements. Similarly, the existing actuators offer a starting point for effecting tradeoff driven adaptation. To be mission-oriented, the tradeoff-driven adaptation mechanism needs also to interface with control elements that can affect mission operations, which usually involves additional instrumentation to be put in the system.

We took the approach that the runtime assessment and tradeoff management must work with the existing system as opposed to designing the survivable system. The analysis of the mission support system and the mission requirements may reveal shortcomings and system owners may decide to address those by modifying the survivability architecture or introducing additional defenses—but the assessment must be with respect to the security and QoS mechanisms that exist within the system, and the tradeoff-driven adaptation must be confined to the control-elements that are available to the system and the missions using it. Therefore, we designed the components supporting the runtime assessment and tradeoff-driven adaptation to augment the existing survivability architecture. Figure 1 shows these components notionally as an overlay. In reality, the runtime control responsible for the assessment and tradeoff analyses is implemented as a distributed agent system.

3.6 Challenges

The central challenge in enabling a mission-oriented runtime assessment is bridging the gap between the IA and QoS requirements of the mission users and the measurements and observables that are obtainable from the system. The other big challenge is to convince mission users that the tradeoff-driven adaptations are actually helpful for fulfilling their mission roles and responsibilities. The latter is not totally a technical challenge, as we observed during our interactions with system owners and mission operators, systems used in real missions are fairly static- all changes must go through a fairly manual and often cumbersome change control process—and the system owners are highly reluctant to accepting dynamic modification done to their system by automated agents. A third challenge is to ensure that the measurements and observations are trustworthy and the computation underlying the assessment and tradeoff management functions are secure and robust against failures and attacks.

3.7 Assumptions

To make the problem of runtime assessment and tradeoff-driven adaptation, we made a number of simplifying assumptions. We already have ideas about relaxing or eliminating some of these simplifying assumptions, and are planning to explore that possibility as follow on work. First, we assume that the mission and system level information and IA and QoS requirements of mission users are available. In other words, we did not investigate ways to automatically extract such information or to define or improve how such information should be provided. We assumed that we will have access to the relevant document and mission users and could simply elicit the information we need by studying the documents or talking to the responsible parties. Second, we assumed that the distributed agent system retains connectivity among the agents and the peers in the system they interface with all the time. The connectivity may degrade, but there is never any loss of connectivity. Third, we assumed that the enterprise security and system management systems exist, and the measurements and observations they collect are readily accessible through enterprise system management (ESM) databases. We also further assume that the data in ESM databases are reliable and trustworthy, i.e., if we learn from the ESM database that CPU of a specific host is overloaded, or a defense mechanism is down—that is indeed so. In other words, the assessment and tradeoff analysis relies on the ESMs for the correctness of measurements, and secure transportation and storage of the collected data. This influences how the assessment and tradeoff analysis components are overlaid into the system—by co-locating the agents responsible for assessment and tradeoff analysis with the ESM databases, we have eliminated the need for securing the ingestion process of measurements and observations into the assessment and tradeoff framework. Similar benefits have been achieved by co-locating the agents with the controllable elements in the system. Finally, we assume that it is acceptable to make modification to the system and mission functions at runtime and the controllable elements to effect such changes exist in the system.

3.8 Research and Development Procedures

We followed a rapid prototyping with spiral improvement approach in this project, where each spiral consisted of a research, design, implementation, and testing and demonstration phases. In the first spiral, we performed a literature study to under the state of the art in security evaluation, followed by a concept development and feasibility study of runtime assessment. This spiral developed the concepts of the multi-dimensional assessment space—the multiple dimensions along which the QoS and security requirements of mission users can be represented for assessment purpose, the metric classes—the generic types of measurements and observations than can be used for runtime assessment, and a high level assessment framework—a template schema for combining the available measurements and observations against user-specified requirements to determine if the system is delivering the required level of QoS and security. We also prototyped the basic data structures and comparison schemas to validate that user requirements and measurements can be used the way envisioned by the assessment framework. In the second spiral we focused on prototyping the agents, called blackboards, responsible for performing the runtime assessment. To test the agents with realistic data that can be varied over time throughout the mission, we developed an emulated network using the Common Open Research Emulator (CORE) [43] and deployed the blackboards on it. In the next spiral we focused on defining the tradeoff space, understanding the interdependency between QoS and security, and identifying what additional information is needed for meaningful QoS-security tradeoff decisions. The ideas were validated against simulated case studies on paper first; and then in part by extending the blackboards and the emulated network to support tradeoff driven actuation. We also conducted preliminary experiments to estimate the time spent on various phases of the tradeoff analysis and actuation steps. These experiments also highlighted the limitations and weaknesses of the initial tradeoff analysis, which were rule based (preferences) and fairly localized and uncoordinated (each blackboard performing its own tradeoff). Informed by this experience, we set out to develop a more sophisticated tradeoff analysis scheme. This led to the formulation of the QoS-security tradeoff as a distributed constraints optimization problem (DCOP). Study of the DCOP formulation and the specific domain (of interdependency in the distributed information system) led to the discovery of a number of specific characteristics of our domain that is different from the traditional domains where DCOP is used. Addressing these idiosyncrasies led to the development of enhancements to a commonly used DCOP solution called max-sum [3]. We first implemented and evaluated the abstract implementation (i.e., stand alone implementation of the algorithm) using canned and synthetic data. In the next spiral, we focused on combining assessment and tradeoff analysis capabilities in the context of a survivability architecture. We used an existing defense-enabled distributed system that included multiple defenses was organized in a survivability architecture to protect the key assets long with standard security measures and ESM monitoring. The system consists of 10s of nodes connected by 4 different networks, implemented using VirtualBox VMs and PFSense virtual routers. We developed a scenario that is roughly emulating how close air support missions are conducted, and can be run on this distributed system. In the final spiral we focused on experimental evaluation and usability aspects. Experiments were conducted to demonstrate runtime assessment and tradeoff-driven actuation and collected performance metrics. Activities included developing a test plan specifying the effects and failures to be injected to the mission scenario and observations and metrics to be collected, preparing and validating the XML configurations for the IA and QoS requirements for the specific run, conducting the experiment which involved running the mission and injecting the failures and attack effects as planned, collecting logs, performing post analysis and documenting results. For demonstration and interactive control of mission execution, visualization components were added to interact with selected mission participants and to display runtime assessment and tradeoff decisions as the mission progresses. As the final step of this phase of the project, we delivered and installed the demonstration prototype to AFRL, and performed live demonstration.

3.9 Expected Impact

The successful result of this project paves a new way to build and run mission critical systems where security and service delivery are *actively co-managed* to *continually maximize the mission utility* throughout its duration. It also demonstrates how *usable information assurance* - instead of the current *all or nothing* and *perception-base* view - can be assessed based on concrete observations and measurements *continually,* and in ways that are *relevant for stakeholders*. Managing information assurance this way and supporting security-QoS tradeoffs will imply a more effective and less wasteful use of the system's services and resources. When attacked, users will still be able to use those parts of the system that are able to meet the acceptable level of service delivery and security. The architecture for managed QoIA will also enable better risk management—instead of guesswork, the impact of operator actions and defensive responses is reflected as changes in the systems operating point on the IA state space.

We envision that future DoD information systems will include a capability that can be thought of as an *always-on risk management service* (see Figure 2 for an illustration), that will continuously assess the level of assurance, indicate when the system is operating below the desired level, and initiate adaptive behavior to accommodate the changes caused by attack effects or by operator behavior within the bounds of the mission objectives. As shown in Figure 2, the continuously updating operational picture and risk assessment can be displayed at a centralized location, but we envision the actual assessment and analysis will be computed in a distributed manner by a collection of QIAAMU agents (shown as QA) located throughout the system. The runtime viewer will present, at any given point within the mission, what QoS and security requirements of the mission stakeholders are being met or not in an intuitively understandable manner (the various panels and the red, green and yellow squares). It will also present the overall status of the mission's assurance state, as shown by the traffic light. In addition, the viewer can display and offer an interface for the human operator to accept or decline adaptation actions (actuations) the tradeoff management algorithm comes up with.

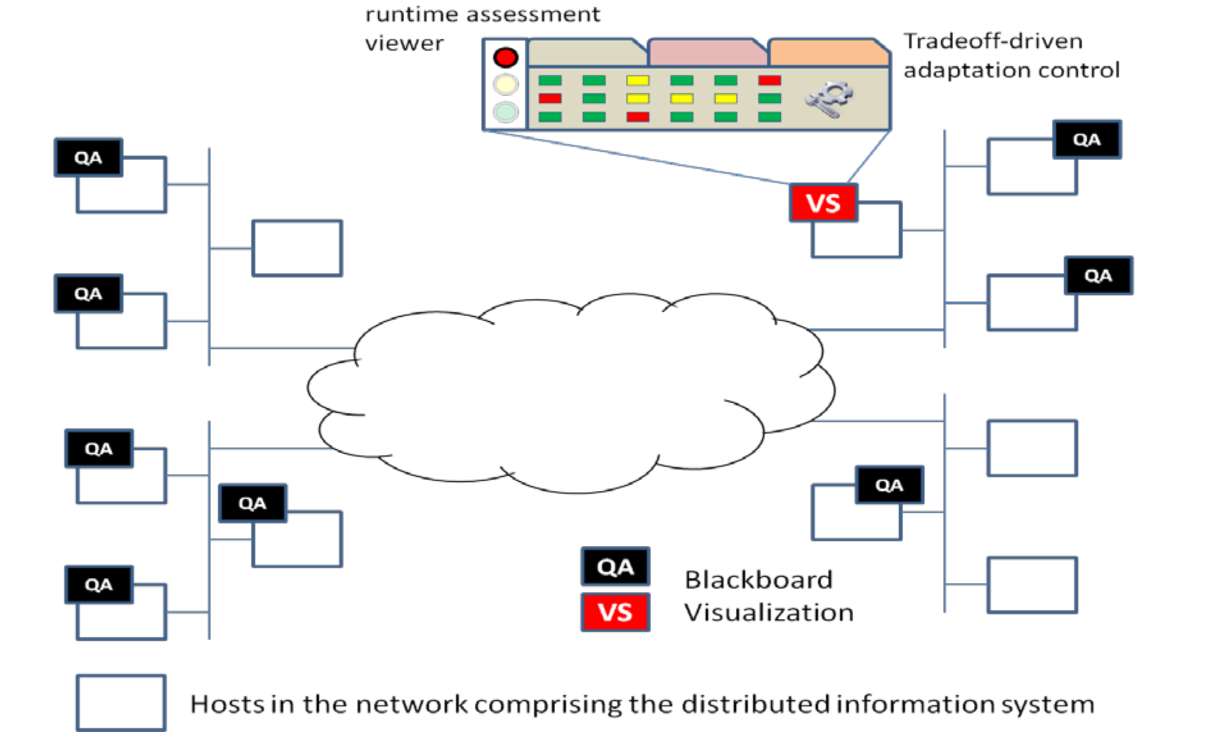

runtime assessment viewer
Tradeoff-driven adaptation control
QA
VS
QA Blackboard
VS Visualization
Hosts in the network comprising the distributed information system

Figure 2: Runtime assessment and tradeoff providing an always-on mission-oriented risk management service

The QIAAMU prototype achieved at the end of this phase of research comes pretty close this vision. QIAAMU provides an easy to understand visualization of met and unmet stakeholder requirements, and an overall assessment of the risk to the mission (red or green). QIAAMU currently does not provide the option for human operators to accept or decline the adaptations that the tradeoff analysis recommends, instead it assumes that QIAAMU can dynamically reconfigure the tunable aspects of the system and simply provides a record of actuations performed.

4.0 RESULTS AND DISCUSSION

This section describes the research findings and technical accomplishments organized in a number of fairly detailed subsections. Section 4.1 covers the formulation and methodology of requirements based mission-oriented IA assessment, defines the assessment space broken down in terms of various attributes and the spatial scope where that attribute is relevant. Section 4.2 describes the metrics, i.e., measurements and observations that can contribute to the runtime mission-oriented assessment. Section 4.3 explains the assessment technique. Section 4.4 describes the distributed tradeoff management algorithm that attempts to maximize the level of satisfaction of the mission requirements by adapting the system settings and configurations. Section 4.5 describes the architecture of the QIAAMU runtime assessment and tradeoff management solution and explains how they fit in with respect to the existing security and survivability architecture of a distributed system. Section 4.6 describes internal test and evaluation. And Section 4.7 provides a case study of using a specific metric class for a special purpose security assessment that researchers at University of Illinois undertook under this project, namely, use of authentication logs to assess the likelihood of account compromise.

4.1 Assessment Space and QoS and IA Requirements

Informed by our past experiences, we decided early on in the project that it is unlikely that a single quantitative measure of security will ever be universally accepted by all practitioners and applicable to all system and mission contexts. Instead, our goal has been to first capture *quantized* levels of IA that are acceptable to the *mission*; and then identify and use observations and measurements that can be collected from the system and its operating environment to determine, on a continuous basis throughout the mission, whether the system is operating within acceptable levels. This is analogous to the approach BBN researchers took in assessing and managing quality of service in distributed systems [1]. We take advantage of the measurements available from existing mechanisms such Enterprise System Management (ESM) or security tools such as OSSEC to the extent possible. Furthermore, as in QoS management, we interface with other IA and resource management mechanisms to access information that is not otherwise visible (e.g., configuration or state information).

A key focus of this work is "mission-orientation." Acceptable levels of assurance in a mission can be viewed as regions of a multi-dimensional state space. The multi-dimensional assurance state space and the methodology to capture "acceptable" levels of assurance are described in Section 4.1.1. We define "continuous assessment" as the process of aggregating and analyzing observations and measurements to determine whether the system is operating at an acceptable level. In Section 4.1.2 we describe the various QoS and IA attributes that can be used in the continuous runtime assessment.

4.1.1 Multiple Dimensions of the Assurance State Space

IA and QoS requirements can change over time during the mission. Therefore, time is a key dimension in the multi-dimensional assurance state space in which continuous assessment takes place. Changes in a mission's assurance requirements can be based on elapsed time or mission events. We support both.

Each mission typically has multiple stakeholders, and each stakeholder typically has his own (time varying) QoS and IA requirements. Therefore, *stakeholder* is the second dimension of the multi-dimensional space. We are initially considering three representative classes of stakeholders: *commander* (with an ownership stake), *warfighter* (end-user stake) and *operator* (system-administration stakes), which represent the 3 possible values in the stakeholder dimension. Figure 3 shows some examples of stakeholder concerns.

Not every stakeholder is interested in the entire system—a stakeholder is usually interested in specific end-to-end capabilities, subsystems (subsets of physical components and networks) and/or subsets of services offered by the system. Therefore, *spatial scope*, capturing the hosts, networks and applications/services that are of interest to a stakeholder is the third dimension of our assurance space. The spatial dimension, expressing the scope of an assessed attribute, is populated by system building blocks at various levels of abstraction. The level of abstraction depends on the stakeholder's concern or interest. System building blocks can be hosts, software services and application, network links, network enclaves or even an end to end interaction. It is possible to describe and understand the structure, topology and dependencies within a system in terms of these building blocks. Therefore the spatial dimension plays a dual role: on the one hand, stakeholders express the *scope* of their QoS or security requirements using the building blocks. And, on the other hand, assurance engineers use the spatial dimension to understand how changes propagate through the system and impact the delivered QoS or security of specific building blocks.

Classically, security of a system is described in terms of *confidentiality* (C), *integrity* (I), and *availability* (A) of the services it provides and/or the information it handles. Availability and confidentiality is defined in terms of *authorized* users, but does not consider the strength of authentication and authorization mechanisms involved i.e., whether it was open access that authorizes any requester (which conflicts with any confidentiality requirement), or authorization was based on a user-provided password or validating a common access card (CAC) (CAC authentication being stronger than password-based authentication). In our exploration we included *authentication/access control level* (A/A) (together, because access rights are usually authorized based on some notion of an authenticated identity) as another security attribute that is operationally relevant but is not covered, rather assumed by C, I and A. Stakeholder's security requirements over time therefore can be described in terms of the spatial scope and the assurance attribute. Similarly, on the QoS side, specifying stakeholder's QoS requirements need both a spatial scope and a QoS attribute such as bandwidth or fidelity. The *attributes* covering security and QoS populate the remaining dimension of our multi-dimensional assurance space.

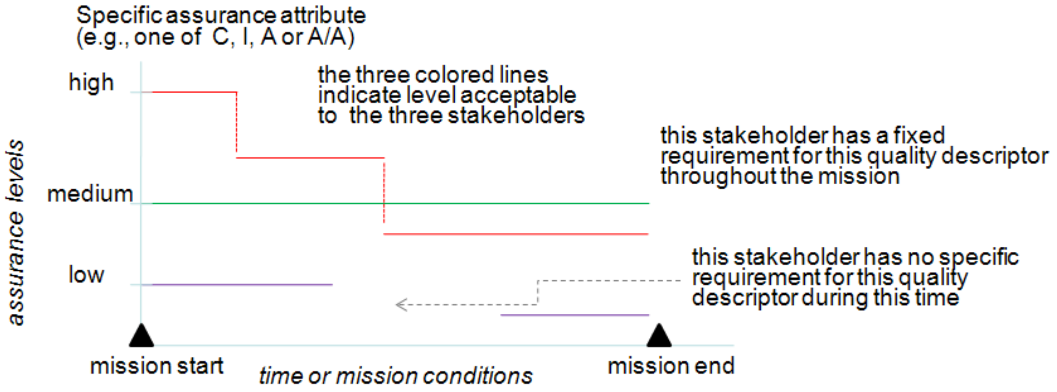


Figure 3: Projection of acceptable levels

It is difficult to visualize the multi-dimensional assurance space. Therefore, we use *projections* to represent and analyze acceptable levels in this high-dimensional assurance space. For example, all stakeholders' interest about a specific security attribute (e.g., C) for a specific spatial scope (e.g., an end-to-end interaction between two applications) over time can be captured in a projection like the one shown in Figure 3. To cover all security attributes for the given spatial scope four sets of such projections will be needed. In Figure 3, the horizontal axis represents mission progression from start to end. Points on the horizontal axis, as mentioned earlier, can denote elapsed time or mission events. Stakeholders have the flexibility to specify their time-varying IA requirement in terms of elapsed time (e.g., for the next 2 hrs since start) or in terms of mission conditions (e.g., from start until an air tasking order (ATO) is published). It is not the case that all projections will have every stakeholder. Furthermore, stakeholders may not have requirements for every point on the time axis.

Note that in Figure 3, the values in the assurance dimension are qualitative and ordered. This is deliberate, because this is the best way to capture stakeholder requirements. Even experts find it hard to describe security quantitatively, and most stakeholders are not security experts. On the other hand, most stakeholders can qualitatively express their time varying C, I, A, and A/A requirements for the system components and services they use or care about. The methodology to capture assurance requirements (in the form of a collection of projections) described in pseudocode, is as follows:

```
For each stakeholder
 For each elem. in his spatial scope
  For each assurance dimension
    Describe in relative terms what is acceptable over time during the mission
```

In the rest of this document we will often overload the term *attribute* to mean *attribute with a scope.* In other words, we may say attribute evaluation to mean evaluating the delivered level of QoS or security attribute for a specific scope.

Apart from the 4 security attributes described above, the QoS attributes we use come from our past work in QoS of distributed systems [1]. In our discussion of attributes, we provide representative examples of measurements and assessment functions to assess individual attributes as motivating examples and establishing that these attributes can be evaluated at runtime for a number of attributes. Not all such attributes may be relevant for a given mission. Later, when we discuss metrics, we will describe representative measurements, the instrumentation or sensors needed to take the measurements, and the underlying causal structure that can be manipulated to change the measurements.

4.1.2 Attributes

In this subsection we describe the general high level attributes that can be used to describe the required and delivered levels of QoS and IA in information systems. These attributes include:

- Availability (A)
- Agility(AG)
- Timeliness(T)
- Fidelity(F)

- Suitability(S)
- Confidentiality I
- Integrity (I)
- Authentication/Access control level (A/A)

Some of these attributes (such as timeliness, fidelity, suitability) are generally only associated with user perceptions of QoS. Some of these attributes (such as confidentiality, integrity, authentication/access control) are generally only associated with user perceptions of security (i.e. assurance). Some of these attributes (agility, availability) are associated with both. We therefore decompose our discussion of these attributes based on whether they primarily affect QoS, security or both. A tabular listing the attributes with an assignment of whether they affect QoS and/or IA can be seen immediately below.

Attribute	**Quality of Service**	**Security**
Availability (A)	X	X
Agility(AG)	X	X
Timeliness(T)	X	
Fidelity(F)	X	
Suitability(S)	X	
Confidentiality (C)		X
Integrity (I)		X
Authentication/Access control level (A/A)		X

It is key to remember that the definition of the attributes—i.e., what they mean is highly customizable depending on the application and mission context. We only provide a general and high level bounding box description. Specific representative examples, guidelines for customizing the attributes depending on application context, and the rationale for selecting these attributes for runtime mission-oriented assessment are discussed in the Appendix A.

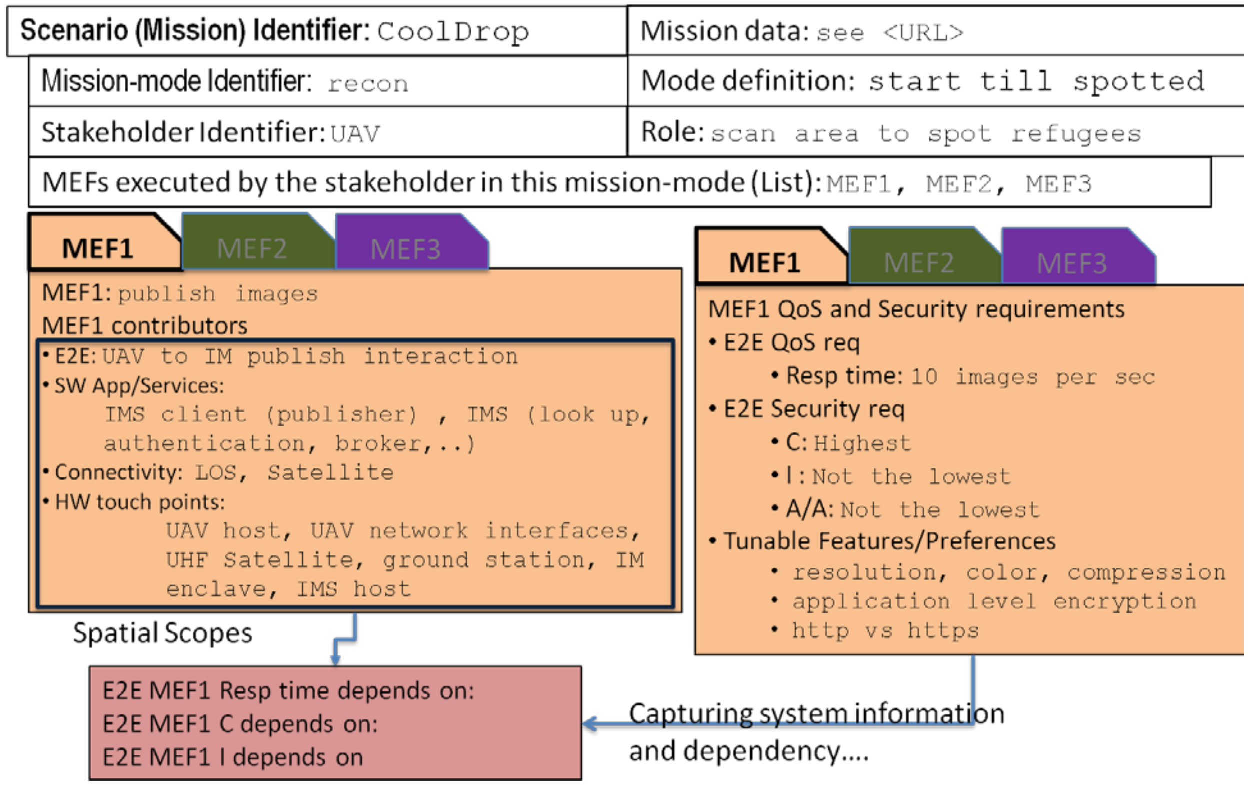
Confidentiality, Integrity and Availability are the three main attributes that are used to describe security. Confidentiality is about preventing disclosure of information to unauthorized individuals or systems. Integrity is the attribute that implies that information or services are not modified in an unauthorized manner. Availability is the attribute that implies information and or services that play a role in providing the information and/or information services are available for use i.e., functioning at a level of acceptable usage. The definition of Confidentiality and Integrity inherently includes the notion of authorization, and authorization is based on authenticating the actors. Therefore the level of Authentication/Access control can also be used to describe the security of an information system or the security requirement of a mission stakeholder. Since adaptation/adaptive response is a key defensive mechanism, how quickly the system can adapt in response to attacks is a good indication of how secure the system is. Therefore we considered Agility as another security attribute.

Availability of the services, especially services that do not have a security function (i.e., performing business functions) is also often used to describe QoS. A service that is sporadically available or highly loaded or takes long to respond is deemed to offer lower levels of QoS. Similarly, Agility, when applied to business functions, i.e., how quickly the system can adapt to heavier demands and overloads can be a QoS indicator. Specifically, a mission stakeholder may have a requirement that response time longer than T can be tolerated only for X seconds, within which response time needs to fall back to less than T.

Timeliness refers to the ability of the information system to respond to service requests in a timely and/or regular manner. In a sense this attributes overlaps the QoS flavor of the Availability attribute. It is not uncommon to have Timeliness as the QoS attribute and use Availability only as a security attribute. Fidelity (also called Accuracy) refers to the “quality” of results returned during service delivery. Fidelity/Accuracy is often confused with Integrity, but they are not the same. Fidelity is the attribute that distinguishes between a low resolution image and a high resolution image of the same object. Integrity is the attribute that asserts that images were not modified during transmission. The final QoS attribute Suitability (also known as Precision) assesses how well the results returned by an information system satisfy requests from the user. When a mission stakeholder requests a weather report for a specific town, but obtains the report for the region the town is in, the attribute that distinguishes the QoS is Suitability/Precision.

4.1.3 Methodology for Collecting Requirements

Now that we have explained the attributes, it should be clear what we mean by specifying QoS and security requirements in terms of these attributes. However, such requirements are usually in the heads of the mission stakeholders or they are in some handbook, explained in a way that is not amenable to automated processing. Therefore, the assurance engineers need to engage in a requirements elicitation process. We envision this to be like any other requirements gathering process, and may involve structured interviews. We developed a requirements elicitation template, shown in Figure 4 to help guide that process. We capture QoS and security requirements starting with a stakeholder S and mission mode M. For this combination, we first identify the mission essential functions (MEFs) executed by the stakeholder in this mission mode. For each mission essential function F used by the stakeholder S in this mission mode M, one can identify the system building blocks that contribute to the realization of F, and its QoS and security. These form the superset of the spatial scopes of all the attributes of mission essential function F that stakeholder S is interested in mission mode M. The QoS and security requirements qualifying stakeholder S's use of mission essential function F in mission mode M are captured in the attributes and their expected qualitative levels for the combination of (S,M,F). This process is repeated for all MEFs the stakeholder S is interested in mission mode M, and then for all mission modes the stakeholder S participates and then the whole process is repeated for all stakeholders. The captured data can then be used by the assurance engineer to formulate and set up runtime attribute evaluations. As we will explain later, the information captured by the elicitation process is used to configure the QIAAMU agents (called blackboards) that are responsible for performing runtime assessment.



Scenario (Mission) Identifier: CoolDrop

Mission data: see <URL>

Mission-mode Identifier: recon

Mode definition: start till spotted

Stakeholder Identifier: UAV

Role: scan area to spot refugees

MEFs executed by the stakeholder in this mission-mode (List): MEF1, MEF2, MEF3

MEF1: publish images

MEF1 contributors

- E2E: UAV to IM publish interaction
- SW App/Services: IMS client (publisher) , IMS (look up, authentication, broker,..)
- Connectivity: LOS, Satellite
- HW touch points: UAV host, UAV network interfaces, UHF Satellite, ground station, IM enclave, IMS host

MEF1 QoS and Security requirements

- E2E QoS req
  - Resp time: 10 images per sec
- E2E Security req
  - C: Highest
  - I: Not the lowest
  - A/A: Not the lowest
- Tunable Features/Preferences
  - resolution, color, compression
  - application level encryption
  - http vs https

E2E MEF1 Resp time depends on:
E2E MEF1 C depends on:
E2E MEF1 I depends on

Figure 4: Template for requirements elicitation

4.2 Metrics Taxonomy

Assessing delivered levels of security and QoS in terms of the various attributes described above requires measurements and observations (i.e., metrics). However, no single collection of metrics is universally applicable to assess attributes in all operational contexts, and it is not feasible to give closed-form general expressions to evaluate the attributes for a given set of measurements corresponding to the metrics. Therefore, we base our discussion in terms of metric classes, and provide several examples of specific metrics from each class that may be objectively measured in a given system.

Organization of security focused metrics in terms of POM, AIQ, EXT, DEF STAT, RES STAT, DEF REP and DEF EFF classes (as shown in Figure 5) and the underlying rationale has been described in prior technical reports [2]. The measurements and observations that are relevant for QoS mirror the structure of the security metric classes. In fact, measurements and observations from POM, AIQ, RES STAT and EXT can be used in QoS management directly. SERV STAT, SERV REP and SERV EFF can be the QoS analogues of the DEF STAT, DEF REP and DEF EFF IA metric classes. One noteworthy difference is that DEF EFF can be a single system-wide measurement (i.e., how effective the defense has been overall, as opposed to measuring the effectiveness of individual defense mechanisms), but SERV EFF measurements, indicating how efficient service has been, is almost always specific to an individual service.

The taxonomy depicted in Figure 5 organizes the metrics based on their nature (e.g., whether they are about the status of defenses or services). It is possible to think of a higher level overlay that organizes the metrics into broader categories as follows:

- Metrics that measure runtime aspects of the system
- Metrics that measure inherent system properties
- Metrics that measure environmental/inorganic operational aspects of the system or mission

In Appendix-B, we discuss specific metrics from various systems organized under these high-level categories, and then show which metric class in the taxonomy they belong to, and how they contribute to the assessment to specific QoS or security attributes.

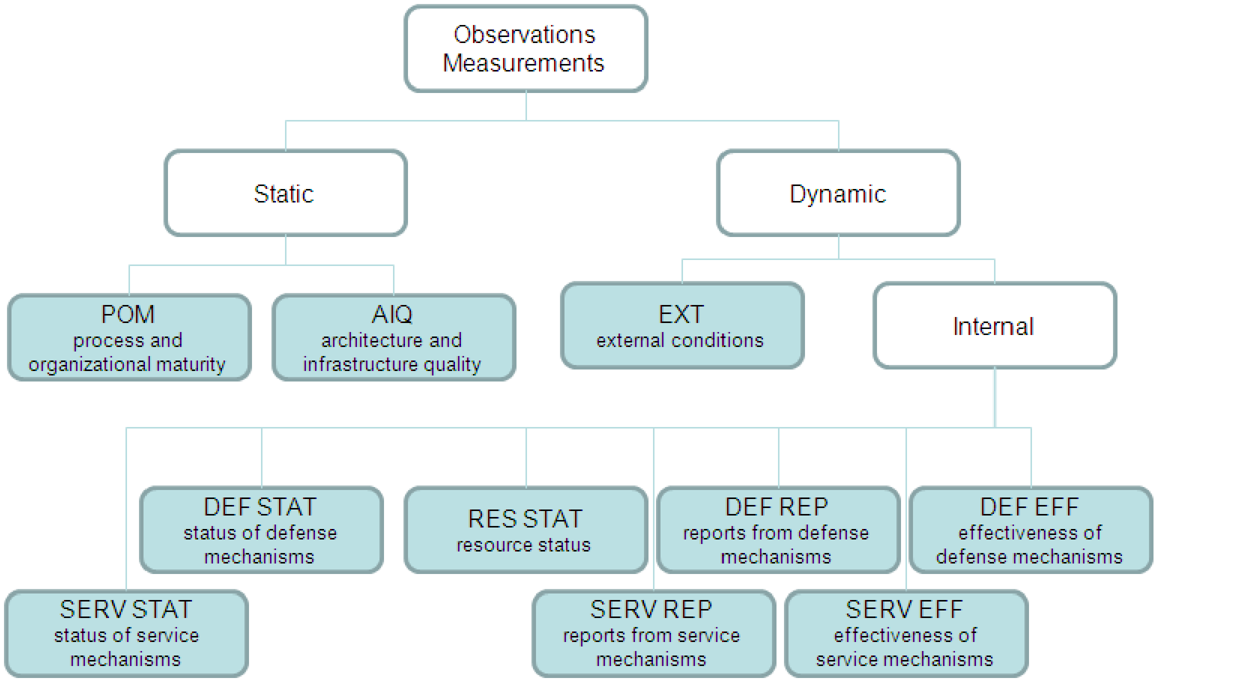

Observations
Measurements
Static
Dynamic
POM
process and organizational maturity
AIQ
architecture and infrastructure quality
EXT
external conditions
Internal
DEF STAT
status of defense mechanisms
RES STAT
resource status
DEF REP
reports from defense mechanisms
DEF EFF
effectiveness of defense mechanisms
SERV STAT
status of service mechanisms
SERV REP
reports from service mechanisms
SERV EFF
effectiveness of service mechanisms

Figure 5: QoS and IA metrics classes

We observe that specific metrics could be associated with multiple attributes. This in turn implies that there is no fundamental binding between a metric class and our QoS and security attributes. In fact, metrics that are relevant for both QoS and IA assessment have been devised or identified by researchers, including ourselves. The table below provides a sample of common metrics i.e., measurements and observations that are generally available in a typical distributed system, along with the metric class they belong to in our taxonomy and the attributes they are associated with. The rows in the table are not quite specific metrics—they need to be instantiated for the specific system and mission context—for example, “service presence” can be instantiated as “presence of the Pub-Sub-Query service”, “protection presence” can be instantiated as “presence of JBoss authentication service”. There may be some overlap in the scope of the measurements and observations depending on operational context i.e., different rows may be (partially) measuring or observing the same physical phenomenon. But we feel that an understanding of these metrics can guide a general approach to understand how QoS or IA level is derived from lower-lever application measurements and observations. The x in green implies indirect impact, whereas a black x implies direct impact.

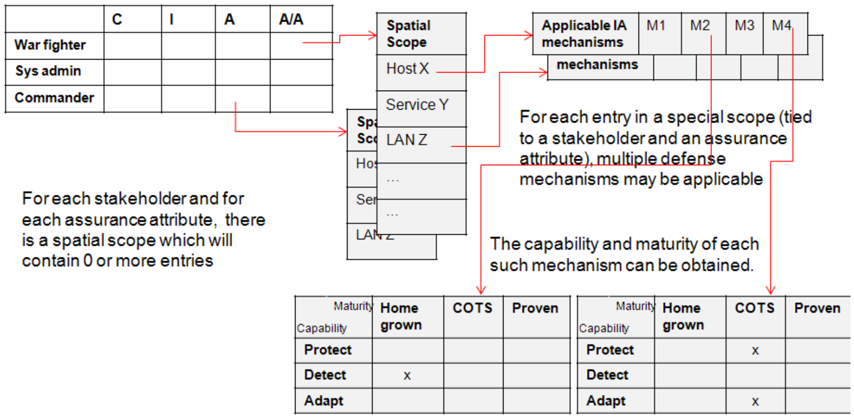
Table 1: Relationship between measurements and attributes

Focus or objective of measurement and observation	Metric Class (Cross ref to Taxonomy)	Availability	Agility	Timeliness	Fidelity	Suitability	Confidentiality	Integrity	Authentication/ Access control level
Service Presence	SERV STAT	X	X						X
Protection Presence	DEF STAT	X	X				X	X	X
Response Time (Services)	SERV EFF	X		X					
Throughput	SERV EFF	X		X					
Accuracy	SERV EFF	X			X	X	X	X	X
Smoothness	SERV EFF				X				
Resource Allocation (utilization)	RES STAT	X	X	X					
Maturity (process & organizational)	POM	X	X				X	X	X
Architecture & Infrastructure Quality	AIQ	X	X	X			X	X	X
Deviation from Configuration (Services)	SERV STAT	X	X	X	X	X			
Deviation from Configuration (Defense)	DEF STAT	X	X				X	X	X
Recent Experience (Services)	SERV REP	X	X	X	X	X	X	X	X
Recent Experience (Defense)	DEF REP	X					X	X	X
Load (e.g., no. of users in the system)	EXT		X	X	X				X
Applicable exploits/adversary expertise	EXT	X		X	X		X	X	X
User Maturity/Expertise/Experience	POM	X	X	X	X	X	X	X	X
Location/Accessibility	DEF STAT	X	X				X		X
Replication Level	DEF STAT	X	X						
Encryption Level	DEF STAT						X		

It should be pretty clear that a metric is not really affiliated with individual attributes (or small sets of attributes), or even dedicatedly affiliated with QoS or security.

4.3 Assessment Methodology

While it is intuitive that measurements from the metric classes influence the level of IA delivered by the system, much like QoS assessment, there is no well accepted all encompassing formula for combining the metrics to determine the delivered level. We have developed mechanisms to determine the delivered level of QoS in our QoS-managed middleware work earlier [1]. We follow a similar approach for IA assessment that we describe in this section.



	C	I	A	A/A
War fighter				
Sys admin				
Commander				

For each stakeholder and for each assurance attribute, there is a spatial scope which will contain 0 or more entries

For each entry in a special scope (tied to a stakeholder and an assurance attribute), multiple defense mechanisms may be applicable

The capability and maturity of each such mechanism can be obtained.

Capability \ Maturity	Home grown	COTS	Proven
Protect			
Detect	x		
Adapt			

Capability \ Maturity	Home grown	COTS	Proven
Protect		x	
Detect			
Adapt		x	

Figure 6: Dependency graph

As in the case of QoS, a number of system analyses, development and integration steps are needed before deployment. For this reason, in addition to the assessment mechanism, we introduce a new software engineering role, namely, the assurance engineer that is analogous to the QoS engineer role, responsible for the underlying requirements gathering, system analysis and instrumentation processes.

However, in QoS engineering, the relations between metrics (e.g., jitter in packet delay, frame rate, CPU load) and attributes (e.g., fidelity of audio or video streams) are typically derived from concretely observable phenomena (e.g., packet jitter over a threshold makes the audio or video quality unusable for the user) and natural laws (e.g., a human eye cannot distinguish beyond a certain frame rate, a high CPU load will increase the response time of the processes running on that host). Variations in delivered IA (e.g., loss of confidentiality) may not be directly visible to the stakeholder, and there is no natural law to assist us. To address this gap, we have developed an assessment *framework* that enables us to relate metrics and attributes in a context sensitive way.

4.3.1 Collecting System Details and Validating Supportable Levels

Given the spatial context and the attribute of interest for each stakeholder-specified requirement, it is possible to determine, with a fair degree of accuracy, the system resources, defense mechanisms and business functions that are involved in servicing the requirement, using a straightforward security focused system analysis. This analysis process is analogous to white boarding in red team methodology, and will produce a dependency graph of the form shown in Figure 6, showing the spatial scope for each stakeholder and for each assurance attribute of interest along with the list of relevant defense mechanisms. The defense mechanisms can be protection-focused, detection-focused, participate in defensive responses (adaptation-focused) or can be any combination thereof.

Let us assume that there are *N* defense mechanisms (D_1, D_2… D_N) that influence security attribute A of the spatial context S that the stakeholder H has an interest in. Typically, these mechanisms are distributed throughout the system, not necessarily local to stakeholder H.

A defense mechanism D_i can be in m_i states ordered in terms of the strength of protection. The states can be as simple as OFF and ON, with the ON state providing stronger protection, but a defense mechanism will typically have more than 2 states. For instance, a firewall can be set to use rule sets with increasing strictness; an encryption mechanism can support 128, 256, 512 and 1024 bit keys. For each D_i, we assign a DEF STAT metric d_i whose value can be integers between 1 and m_i, where 1 represents the setting providing the weakest protection and m_i the strongest.

The configuration of the defense mechanisms relevant for A in S for H can therefore be represented as a vector of DEF STAT metrics [d_1,d_2,…d_N]. The possible values of this vector forms a partially ordered lattice with [m_1, m_2… m_N] at the top and the unit vector of size N at the bottom. Some combinations of defense mechanism states are invalid. For example, if one DEF STAT value indicates that one side of S is set up to encrypt and another DEF STAT indicates that the other side is set up to handle clear text, this configuration is not workable. Similarly, some combinations may not offer different levels of assurance. Therefore the lattice will have some invalid nodes that we exclude, and some equivalent nodes that we treat as a group. After eliminating the invalids and grouping the equivalents, we partition the lattice into the number of levels that the stakeholder specified such that the nodes in the upper part of the lattice correspond to the higher levels of assurance, a process we call *banding*. It is possible that the required number of levels exceeds the number of possible bands, in which case the assurance engineer reconciles this conflict with the stakeholders.

This process is illustrated in Figure 7 with 4 defense mechanisms, where d_1 and d_3 are binary valued (they can be either 1 or 2), and d_2 and d_4 have values in the range [1, 3]. Invalid configurations are shown by the red boxes in Figure 1, indicating configurations where (d_1= 1 and d_4= 2) or (d_1= 2 and d_1= 3) are incompatible. Equivalent configurations are shown in Figure 1 by the dotted red shape, indicating that the configurations where (d_1 = 1, d_2 = 1, d_3 = 1, d_4= 3) and (d_1= 1, d_2= 1, d3= 1 and d_4 = 2) are effectively the same. Equivalent configurations need to be banded together (as shown in Figure 7, the equivalent configurations all belong to the blue band.) The figure also shows the banding for 3 required levels. Note that the invalids and equivalents can appear anywhere in the lattice, but to be meaningful, bands should cluster nodes that are not far from each other (e.g., putting (1,1,1,1) and (2,3,2,3) in the same band would imply that the weakest and strongest configurations offer the same level of assurance.)

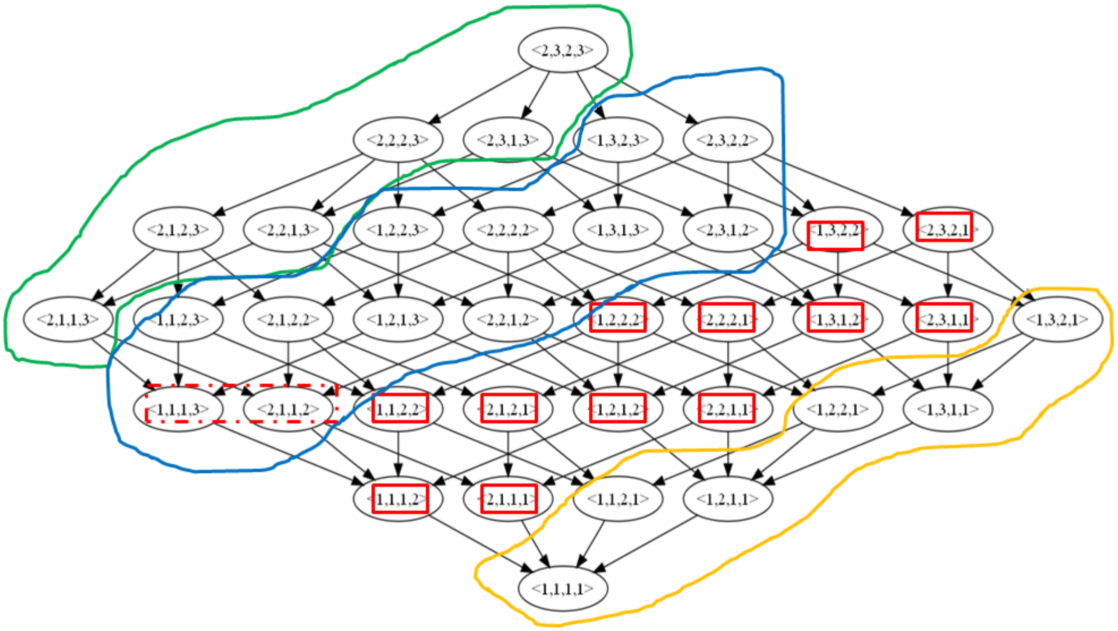


Figure 7: State space of DEF STAT and banding

4.3.2 Base Assessment Scheme

In our proposed framework, delivered level of assurance $L_{A,S\ H}$ for an attribute A and spatial context S the stakeholder H is interested in (we will use *A in S for H* as a shorthand henceforth) at a given time equals V ($B_{A,S,H}$) where $B_{A,S,H}$ is a *base level assessment* and V is a *variance* function. Only the DEF STAT and RES STAT metrics, which are easily available in most modern systems, are mandatory in this framework. All other metrics are optional to accommodate the fact that these measurements may not be available in some systems. The base level $B_{A,S,H}$ is assessed by establishing a mapping from the collection of DEF STAT values relevant for S to the desired number of levels the stakeholder used to express his requirements. This process is described next. See Section 4.3.2.1 for a case study of using DEF REP metrics in assessment.

The algorithm to assess the base level $B_{A,S,H}$ (which will also be the assessed level in the absence of the variance function) then becomes membership checking: which band contains the current configuration indicated by the observed DEF STAT values? But explicit membership checking on a large graph can be time consuming (the total number of nodes in the graph is the product of m_i s, and can be extremely large). To address that issue, we make use of the atLeastAsSecureAs relation, which captures the partial ordering implicit in the DEF STAT space, between two nodes defined as follows:

```
atLeastAsSecureAs([],[]).
atLeastAsSecureAs([F1|R1],[F2|R2]):-
     F1 >= F2,
     atLeastAsSecureAs(R1,R2).
```

In the above code fragment, a node N is defined as a list of values (e.g., N = [a, b, c, d] and the [F | R] notation implies F is the first element in the list and r the remainder of the list (e.g., F= a and R = [b, c, d] for the list [a, b, c, d]). It is straightforward to define the various colored bands in terms of the invalid, equivalent and atLeastAsSecureAs relations.

The variance function V may not be applicable to all assessments. More specifically, the mandatory RES STAT measurements modify the base level assessment of availability only. The DEF REP, DEF EFF and EXT measurements, when available are used to modify the base level assessment of confidentiality, integrity and access control/authentication strength. The POM and AIQ measurements, when available, are used to reduce the impact of EXT measurements on the assessed value. Although somewhat intuitive (e.g., many alerts about the confidentiality of a link should cause a downward revision of all assessments that have the link within its spatial context, and confidentiality as the IA attribute), the variance function is highly complex. Difficulties include identifying which assessments are to be modified when a DEF REP or EXT measurement changes, the extent of damping if a POM or AIQ value is available, and the extent and direction of revision. We are actively working on expanding these topics; see Section 4.4.3 for guidelines that the assurance engineers can follow to determine the variance functions on a case by case basis.

4.3.2.1 A Case Study

Consider the following hypothetical scenario (schematically shown in Figure 8) derived from a military application where a blue unit uses a mobile platform (laptop plus radio and SATCOM gear) for information operations such as calling in air support or feeding air operation centers with ground intelligence. The blue unit can use SATCOM to connect with a ground-based (enterprise) information management service (IMS) or it can take advantage of a short distance directional link to connect with (a tactical) IMS on board an airborne asset that is in range. The SATCOM link is slow but more secure, whereas the directional link is more vulnerable to attacks from the red forces in the vicinity. The airborne asset always maintains a secure channel with the ground station. The airborne and ground-based enclaves run their own management server (a management server for the man packable client enclave is not necessary), and each enclave runs its own aggregator node. Aggregator nodes can obtain observations and measurements directly from local System Conditions (SCs) (if any), from local system management station (if there is one on the same host) or from a peer aggregator node.

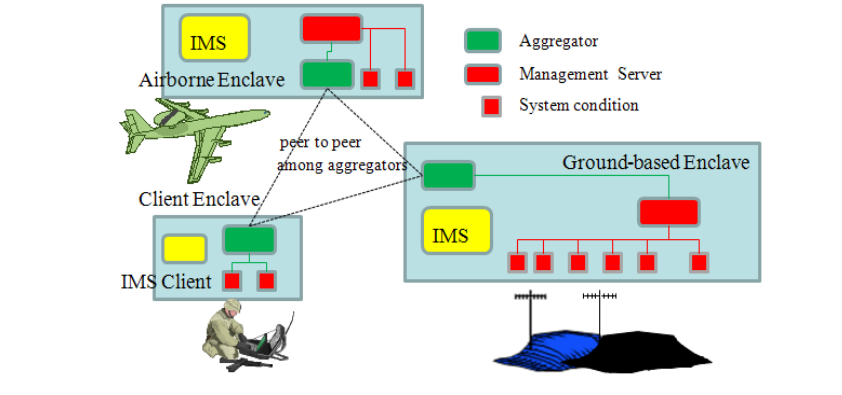
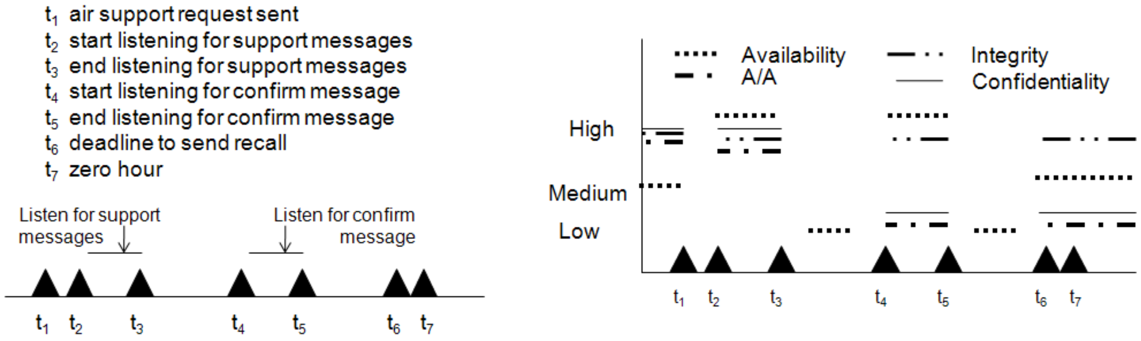


Figure 8: Example scenario for the case study

For the case study, we focus on the blue unit end user as the stakeholder (H), who is primarily interested in his ability to interact with the IMS. So the spatial scope (E) is IMS interaction (end to end). Figure 9 (left) shows the mission epochs for the blue force client (described in terms of mission events), and Figure 9 (right) shows the projection capturing his IA requirements throughout the mission. The stakeholders availability requirement is expressed in terms of three levels, whereas C, I and A/A requirements are expressed in terms of two levels. There are periods in the mission when the stakeholder only cares about availability. For brevity, and to reflect the currently implemented capability, we limit our discussion primarily to DEF STAT and RES STAT SCs.



t1 air support request sent
t2 start listening for support messages
t3 end listening for support messages
t4 start listening for confirm message
t5 end listening for confirm message
t6 deadline to send recall
t7 zero hour
Listen for support messages
Listen for confirm message
t1 t2 t3 t4 t5 t6 t7
Availability
A/A
Integrity
Confidentiality
High
Medium
Low
t1 t2 t3 t4 t5 t6 t7

Figure 9: Stakeholder requirements

The relevant IA mechanisms of this system include the redundancy offered by the airborne and the ground enclave hosting the IMSs, network level encrypting devices at the blue unit as well as the IMS enclaves so that the blue unit has the option to communicate securely over encrypted channels. In addition, the IMSs offer two levels of service: in one the user needs to log in, and in the other he does not- think of them as different URLs or service endpoints, one accepts what is sent, but expects a HMAC or some other token based on shared secret; the other redirects to a login page. In a fly by situation, the blue unit uses the former because there may not be any time to log in. In general, if there are k defense mechanisms relevant for a particular assurance attribute A in the context of a specific spatial scope E of a specific stakeholder H, we will have n system conditions where n>= k. This is because some of the defense mechanisms have distributed components at multiple locations, and DEF STAT information from such mechanisms will come from different parts of the system. The DEF STAT and RES STAT system conditions for our scenario are enumerated in Table 2 (1st column is id with short description 2nd column is what they report.

Table 2: Relevant DEF and RES STAT system conditions

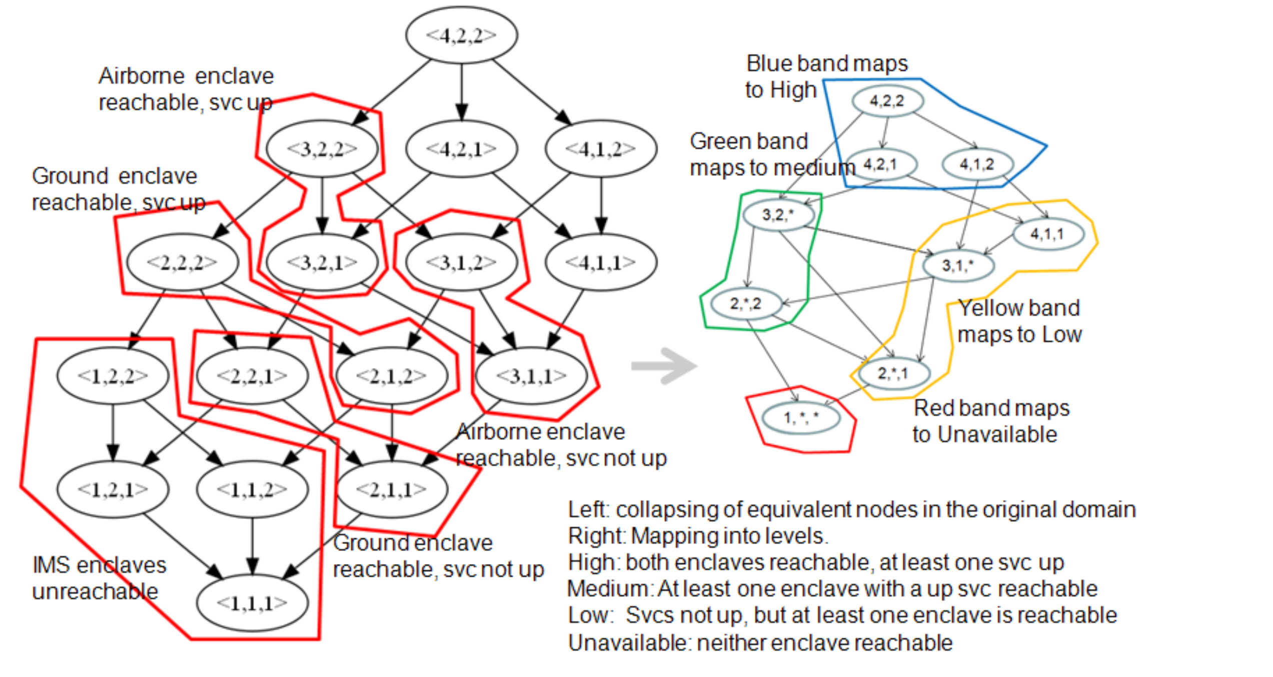
DEF STAT at client enclave	
D_1: IMS_Host Reachability	1: none, 2: Airborne 3:Ground-based, 4: both
D_2: SATCOM. scrambler	1: OFF, 2: ON
D_3: Directional scrambler	1: OFF, 2: ON
RES STAT at client enclave	
R_1: SATCOM bandwidth	In Kbps
R_2: Directional bandwidth	In Kbps
DEF STAT at ground-based enclave	
D_4: SATCOM scrambler	1: OFF, 2: ON
D_5: Firewall status	1: OFF (stopped) 2: ON
D_6: IMS_open (w/HMAC)	1: Not offered 2: Offered
D_7: IMS_auth (w/log in)	1: No offered 2: Offered
RES STAT at ground-based enclave	
R_3: IMS Host load	[1..100] 1=min load, 100 max load
DEF STAT at airborne enclave	
D_8: Directional scrambler	1: OFF, 2: ON
D_9: Firewall status	1: OFF (stopped) 2: ON
D_{10}: IMS_open (w/HMAC)	1: Not offered 2: Offered
D_{11}: IMS_auth (w/log in)	1: No offered 2: Offered
RES STAT at airborne enclave	
R_4: IMS Host load	[1..100] 1=min load, 100 max load

The domain of the baseline function $B(\cdot)$ for various assurance attributes are shown in Table 3. D_{12}, noted within parenthesis, is a new SC indicating the currently used enclave(s) (1 airborne, 2 ground-based, 3 both) and acts as a selector in $B(\cdot)$ for C, I and A/A, determining the DEF STAT SC subset to be used. The selector is not applicable to A, because it, by definition needs to consider availability of both IMS enclaves.

Table 3: Domain of various baseline assessment functions

$B_c(\cdot)$	D_2, D_3, D_4, D_8 (and D_{12})
$B_I(\cdot)$	D_2, D_3, D_4, D_8 (and D_{12})
$B_A(\cdot)$	D_1,D_6,D_7,D_{10}, D_{11},
$B_{A/A}(\cdot)$	D_5, D_6, D_7, D_9, D_{10}, D_{11} (and D_{12})

Each DEF STAT system condition D_i can have values in the range [1, x_i], where the higher values imply stronger security, unless explicitly stated to be equivalent. For example, D_1=2 and D_1=3 do not really reflect different levels of availability, so they are equivalent in our scenario. Given this, the domain space of each B(·) can be described as a Hasse diagram, showing the partial ordering of tuples of various possible values of the system conditions as nodes. The Hasse diagram for $B_A(\cdot)$ with 64 nodes of 5-tuples is too complex to explain, so in Figure 10 (left) we show a Hasse diagram with D_1, D_7 and D_{10} (values in the nodes are in that order), which would be the case if the airborne enclave offered only the open IMS and the ground-based enclave offered only the IMS that requires explicit log in. Under the same assumption the domain space of $B_{A/A}(\cdot)$ will involve 4-tuples with values of D_5, D_7, D_9 and D_{10} (in that order). Not all nodes indicate meaningful states in the system because of physical constraints. For example, D_1 = 1 implies none of the IMS enclaves are reachable, so all states with D_1 = 1 but D_7 or D_{10} = 2 is meaningless to this stakeholder's availability requirement. Similarly, D_1 = 2 or 3 implies only one (airborne or ground) enclave is reachable, so the state of the IMS in the other (ground or airborne, respectively) is immaterial. Besides, implicit relationship between the system conditions that reflect observations at different end points of individual defense mechanisms that offer end-to-end capability (such as encryption) imply that some nodes in the domain space will be *invalid*.



Left: collapsing of equivalent nodes in the original domain
Right: Mapping into levels.
High: both enclaves reachable, at least one svc up
Medium: At least one enclave with a up svc reachable
Low: Svcs not up, but at least one enclave is reachable
Unavailable: neither enclave reachable

Figure 10: Banding of DEF STAT state space

The red shapes in Figure 10 (left) indicate collapsing of equivalent nodes. The reduced Hasse diagram and grouping of nodes into bands that map to the three levels of availability as required by the user is shown in Figure 10 (right). Similar exercise for all B(·) will lead to the mapping from DEF STAT to the qualitative assurance levels specified by the stakeholder. Without going into the details, let us simply state that if the airborne enclave is used i.e., D_{12} =1, mapping of $B_{A/A}(\cdot)$ can be defined by the banding of 2-tuples representing values of D_9 and D_{11}: (2, 2) = High, (1, 2) = Low, and declaring all others as invalid. If the ground enclave is used, i.e., D_{12} =2, the mapping will be in terms of D_5 and D_7. If both enclaves are used (i.e., D_{12} = 3), it can be defined by banding 4-tuples representing values of D_9, D_{11}, D_5 and D_7 (in that order): (2, 2, 2, 2) = high, (2, 1, 2, 1) = low, and all others invalid. It should be obvious that the system can support the number of levels required by the stakeholder for C, I, A and A/A. Once the equivalent states are collapsed, invalid states are noted and banding of nodes to the levels are done, determining the level of assurance at any given time in the mission reduces to membership checking for groups of system condition values—such as given current values of D_1, D_7 and D_{10}, which band does it belong to?

Table 4: Composing base and variance functions

B(.)	g(.) = NW load < 60% **and** Host Load < 60%	Assessed V(B(.))
Blue	1	High
Blue	0	Medium
Green	1	Medium
Green	0	Low
Yellow	1	Low
Yellow	0	Unavailable

Using only RES STAT for variance, the variance modification is typically applied to the assurance attribute *availability* (i.e., for others, the variance amounts to identity function). Continuing with the case study, it is possible to define availability assessment i.e., the $L_{H,E,A}(\cdot,\cdot)$ when A= availability, is a mapping from the system condition values to the 3 levels required by the stakeholder as shown in Table 4. In Table 4, the 1st column shows the baseline assessment (i.e., B(·)) based on the observed state of the DEF STAT SCs. Column 2 shows a logical AND of two composite RES STAT system condition values, one representing the load of the network between the service user and the service provider (a selection or a composition of R1 and R2, depending on which enclave(s) is (are) being used) and the other representing the load at the server (Host load) (once again, a selection or a composition of R3 and R4 depending on which IMS(s) is (are) being used). The definition of load is deliberately kept unspecified—it is up to the assurance engineer to use a quantity that is indicative of the network or system load. Existing system management mechanisms routinely provide a composite load indicator for the network, host platform and for application services; the assessment mechanism can accept any of these values. Column 3 shows the overall assessed availability value (i.e.,$L_{H,E,A}(\cdot,\cdot)$) for stakeholder H for the spatial scope E.

Let us now consider a time when the blue unit is about to call in air support. In the absence of airborne asset, they have just connected to the ground enclave. But, after obtaining the SC values, the aggregator node at the blue enclave indicated that the assessed availability is “Medium” (which is what they require at that point), due to heavy network and host load. This enables the blue unit to make a well-informed decision whether to wait or to continue at the risk that in the next mission epoch the system may not be able to deliver the required availability. Without the CMA instrumentation and assessment infrastructure, the war fighter would have to spend considerable efforts to obtain similar information. Next, assume that an airborne asset became reachable after a while, making the availability level “High”. The blue unit user then establishes a secure channel to the airborne enclave and sends the air support request to the open IMS (including the security token in the message as required), entering into the critical mission epoch where C, I, A and A/A all are required to be “High”. Let us now introduce an attack (perhaps the enemy forces in the region sensed the radio communication and attacked the airborne enclave), which resulted in degradation of the link to the airborne asset and crashed its firewall, which will be reflected by the A and A/A levels falling below the required “High”. This will prompt the blue unit to be cautious about interacting with the airborne asset, establish a secure link with the ground station, and log in with the IMS there. If we consider a system administrator at the ground enclave as a new stakeholder, the events communicated from the blue enclave to the ground enclave will provide meaningful information to assess the IA level he is interested in (in addition to, or in spite of, depending on the how the attack impacts the airborne asset, the measurements and observations that may come from the airborne enclave directly). Without CMA, both stakeholders will continue to use the system unaware of the attacks, until it is too late. As a final illustration consider a different attack impact, where a host/application that is not part of the spatial scope of the blue unit client in the airborne enclave is taken out. This may lead to unacceptable levels of IA for the system administrator stakeholder at the airborne enclave, but the aggregator node will show that blue unit client can still obtain the required level through the airborne asset, if nothing else is available. Without CMA, this would deem the airborne enclave unsafe to use and hinder the tasks of the blue unit.

4.4 QoS-Security Tradeoff

Almost every aspect of our life today depends on distributed information systems, and the frequency and sophistication of cyber attacks targeting these systems is on the rise. It is no surprise that IA is a growing concern for the DoD, since the US military depends on information technology to conduct national security and warfare more than any other nation in the world. However, IA cannot be considered in isolation, as it interacts with Quality of Service (QoS). In the presence of limited resources, the security mechanisms employed for IA (e.g., firewalls, antivirus, encryption, etc.) usually adversely affect QoS levels delivered by a system. The system therefore needs to make a tradeoff between IA and QoS. This tradeoff is further complicated by the fact that users' relative preferences over QoS/IA aspects change based on the situation, the preferences of different users conflict, and tradeoff decisions made at one node in the distributed system typically affect other nodes as well.

In the QIAAMU project, we designed and implemented a distributed computation which, when carried out by the nodes in a distributed system, brings the delivered levels of QoS and IA as close as possible to the levels required by the user. Specifically, the nodes in the system make coordinated decisions as to what local actions to take to optimize QoS/IA levels delivered by the system. Our first contribution is formulating this problem as a Distributed Constraint Optimization Problem (DCOP). This entails quantifying various aspects of the system in order to be able to compare options in the course of optimization, as well as encoding the effects of various decisions on the quantities we want to optimize. The DCOPs we obtain have cost functions where multiple local configurations result in the same cost. In addition, the corresponding factor graphs contain many cycles. To deal with these issues, our second contribution is a value propagation phase that helps distributed nodes to reach a consistent set of decisions even in cyclic factor graphs with non-unique local minimizers.

4.4.1 Tradeoff Background

Multiple security mechanisms are increasingly being used to defend modern distributed information systems against malicious cyber attacks. Information Assurance (IA) is concerned with ensuring that the security mechanisms are effective, and the system can be entrusted with critical information processing tasks. Traditionally, IA has taken an all-or-nothing approach where the entire system is deemed secure or insecure, mostly depending on the user's perception. This approach results in lost opportunity when a system is declared insecure and all activities are halted, but in reality only parts of the system have been compromised and some activities could have progressed without any problem. This approach also pushes the users to take undue risk, because there are times when the user must act, and even though their actions (e.g. installing software or disabling a security mechanism) may diminish the overall security of the system, the impact is not reflected in the security state perceived by other users.

In the QIAAMU project we advocated runtime assessment of IA where variation of the system's level of assurance due to attack-induced failures, environmental threats (e.g., release of a new virus), and user-made changes are taken into consideration. Runtime assessment also reflects the fact that security requirements themselves are changing and depend on where the system is within the mission its users are trying to accomplish.

Security mechanisms use the same system resources (e.g., CPU, memory, network bandwidth) that are needed by the information system they aim to defend. As a result, IA and Quality of Service (QoS) interfere, often adversely. For example, increasing the strength of encryption consumes more CPU resources and increases the round trip response time for service request and response, thereby negatively affecting the availability and timeliness, QoS levels, delivered by the system. In mission-critical applications of distributed information systems (e.g., network centric warfare, telemedicine, internet voice/video applications), degraded QoS may mean loss of service, with impacts ranging from loss of revenue to loss of life.

This contention over resources necessitates a tradeoff between security and QoS delivered by a system. There are three complicating factors in this tradeoff. The first is that that there are typically multiple users (stakeholders) of the system. These stakeholders can have conflicting requirements and preferences. Left to themselves, the competing and conflicting requirements of different stakeholders, even when they are participating in the same mission, can result in degraded performance where less important requirements are met at the cost of more important ones. The second factor is that an individual user's relative preferences for QoS and security are not static, but depend on where he is within the mission he is trying to accomplish. For example, an intelligence analyst may prefer to have high definition video, but if there is an external threat, he can make do with standard definition if it is over an encrypted channel. There will also be situations where one aspect of QoS (or security) is preferred over another. For example, a black and white video with low frame drop rate may be preferable to a color video with dropped frames. The third complicating factor is that tradeoff decisions made at one node in the distributed system can affect QoS and IA levels at other nodes. For example, the decision to use a low bandwidth encrypted channel may affect the bandwidth available to other services communicating over this channel.

We developed algorithms and supporting infrastructure to enable user-specific requirement-based runtime management of QoS and security in a unified way. Our *Continuous Mission-oriented Assessment* approach relies on existing security and system management infrastructure to collect measurements and observations to assess whether the delivered levels of assurance, covering both service delivery (i.e., QoS) and security (i.e. IA) meet the user's requirements. Complementing the assessment, runtime management of QoS and IA also needs to take or suggest remedial actions when there are not enough resources to meet all QoS and IA requirements of all users. We address the problem of distributed tradeoffs in this context. More specifically, we calculate tradeoffs between various aspects of QoS and IA in a way that maximizes the satisfaction of all stakeholders. For each decision making node in the distributed system, the decision problem we consider is: what local actions must be taken to optimize service delivery and security provided by the system as a whole.

Our work in this context makes two major contributions. First, we formulate the decision making problem as a Distributed Constraint Optimization Problem (DCOP). To do this, we first quantify various aspects of the system in order to be able to compare options in the course of optimization. We also need to somehow encode the effects of the various decisions on the quantities we want to optimize. In our case, the effects are more complex than the constraint functions generally used in DCOP literature. Our second contribution is a message exchange phase to be used after utility propagation (e.g. using the max-sum algorithm) whose goal is to handle the special characteristics of the DCOPs resulting from QoS and IA tradeoff scenarios. Our DCOPs have the following characteristics: 1) the corresponding factor graphs contain many cycles of various lengths, 2) some nodes have large degrees, 3) multiple configurations of variables minimize any given function, potentially resulting in locally optimal, but globally inconsistent choices. We propose a value propagation phase that helps nodes reach a consistent set of decisions in spite of the cyclic nature of the graph and the non-uniqueness of decisions that optimize the QoS/IA levels at any one node.

The rest of this section is organized as follows: we first briefly review DCOPs and the max-sum algorithm, and then describe in detail how we formulated the distributed tradeoff problem as a DCOP. Next we present our modifications to max-sum and our value propagation phase, followed by the results of our max-sum variants and the comparison to other DCOP algorithms. Finally, we conclude the section highlighting the directions for future work.

4.4.2 DCOP and Max-Sum Background

In a Distributed Constraint Optimization Problem (DCOP), we have a set of n variables $\mathbf{x} = \{x_1, x_2, ..x_n\}$ and a corresponding set of finite domains $\{D_1, D_2, ..., D_n\}$. There is a set of m constraint functions (constraints) where each function involves a subset of variables (its *scope*) and maps each configuration of these variables to a cost. A *local* assignment for function f_i is an assignment of values to variables in its scope. A *global* assignment is an assignment to all variables in $\mathbf{x}$. Because we formulated our problem as a minimization problem, the goal is to find a global assignment that minimizes the global objective function, usually a summation of the constraint functions. Formally, the goal is to find a global assignment $\mathbf{x}^*$such that

$$\mathbf{x}^* = \arg\min_{x} \sum_{i=1}^{m} f_i(\mathbf{x_i})$$

where each f_i is a constraint function and $\mathbf{x_i}$s are the values of variables in its scope. There may be more than one assignment that minimizes the global objective function.

4.4.2.1 The Max-Sum Algorithm

The max-sum algorithm is a message-passing algorithm that has been widely used to solve DCOPs [3]. It is a member of a family of algorithms relying on the Generalized Distributive Law. Max-sum operates on a factor graph representation of the DCOP where there is a function node for each constraint function and a variable node for each variable. There is an edge between a variable $\mathbf{x_i}$and a function f_j if the former is in the scope of the latter. Nodes in the factor graph exchange two kinds of messages:

- Message from a variable to a function

  $$q_{i\to j}(x_i) = \alpha_{i,j} + \sum_{k\in\mathcal{M}_i\setminus j} r_{k\to i}(x_i)$$

  where $\mathcal{M}_i$ is the set of functions that are neighbors of variable x_i in the factor graph, and $\alpha_{i,j}$ is a normalizing constant that prevents the values in messages from increasing indefinitely in a cyclic factor graph.

- Message from a function to a variable

  $$r_{j\to i}(x_i) = \min_{\mathbf{x_j}\setminus i}\left[f_j(x_j) + \sum_{k\in\mathcal{N}_j\setminus i} q_{k\to j}(x_k)\right]$$

  where $\mathcal{N}_j$ is the set of variables that are neighbors of functions f_j in the factor graph.

If the factor graph is a tree, max-sum is guaranteed to converge on the optimal value assignment. Moreover, it can find this assignment in only two sweeps of the tree. The first sweep proceeds from the leaves of the tree upward, with each node getting messages of one of the types listed above and sending messages of the other type. In the second sweep, each variable node x_i locally determines its optimal value by calculating the following function from the messages it received:

$$z_i(x_i) = \sum_{j\in\mathcal{M}_i} r_{j\to i}(x_i)$$

and setting itself to the value that minimizes this function.

One of the attractions of max-sum is that the size of the messages is typically small; message size scales with the size of the domains, as opposed to some algorithms where message size is exponential. However, max-sum is not guaranteed to converge in cyclic graphs, and when it does, the solution it finds is not guaranteed to be optimal. Nevertheless, empirical results show that it performs well on cyclic graphs.

4.4.2.2 Other algorithms

As we show in Section 4.4.3, our DCOPs have factor graphs that contain cycles, and their constraint functions have non-unique minimizers (the minimum of Eqn 1) is attained at more than one value. The basic max-sum algorithm has no special provisions for either of these two issues, so the values chosen according to Eqn 1 may give rise to an inconsistent global solution. In the following paragraphs, we mention some algorithms or variants of max-sum that are concerned with these issues.

The issue of graphs with cycles is addressed by Distributed Pseudotree Optimization Procedure (DPOP) [4], which is a follow up for an older algorithm, DTREE [5] that is only correct for trees. In a tree, the utility reported by a sender node to its parent only depends on the subtree rooted at that node. In a graph with cycles, this is no longer the case. In the pseudotree that DPOP creates from the graph, this utility may depend on one or more variables *above* the parent that are connected to the sender by *back edges*. The UTIL messages exchanged in DTREE are therefore not adequate, since they only report the optimal utility for the subtree for each value of the parent. DPOP's UTIL messages report utility for each value combination of *all* parents of a node; the parent through a tree path as well as parents on back edges. Non-leaf nodes then proceed to combine and pass along UTIL messages up the tree. Value propagation is largely similar to that of DTREE.

DPOP is optimal, even on general graphs. Unfortunately, although the number of messages exchanged by DPOP is linear in the size of the tree, the size of a UTIL message can be exponential in the induced width of the pseudotree. There have been several variants of DPOP that try to address the message size issue [6].

Another approach to dealing with cycles is to remove them. Bounded max-sum [7] is an algorithm that removes cycles by eliminating dependencies between functions and variables which have the least impact on the solution quality and subsequently uses max-sum on the resulting spanning tree. The result is a bounded approximation of the optimal solution of the original problem. However, when constraints are not binary (involve more than two variables), the algorithm's choice of which dependencies to eliminate may not be the one that minimizes the impact on solution quality. As we state in Section 4, the constraint functions in our DCOPs have a large *arity*, so it is not clear that this approach will work well in our domain.

Our value propagation phase is somewhat similar to value propagation in Max-Sum-AD [8]. In that work, the cycles in the factor graph are addressed by assuming an order over the nodes and allowing message to flow in only one direction according to the order. After convergence in one direction, the algorithm proceeds in the opposite direction. Breaking ties and dealing with inconsistent assignments is addressed using a value propagation phase. Experimental results are only reported for binary constraint functions. We plan to compare the two approaches to value propagation on problems with functions of large *arity*.

The work on Multi-Objective DCOPs [9] also deals with cyclic graphs with non-unique minimizers. In this case, the non-uniqueness is due to the presence of multiple objective functions, which means that there may be multiple Pareto optimal assignments that do not dominate each other. The cycles are addressed by bounded max-sum. The non-uniqueness is addressed by a value propagation phase executed on the cycle-free factor graph computed by the bounding approach.

In summary, cyclic graphs can be dealt with by having a sophisticated utility propagation phase and a relatively simple value propagation phase (DPOP), or having a simple utility propagation phase but a sophisticated pre-processing of the graph into a tree (bounded max-sum). As we detail next, our approach is to do neither pre-processing nor sophisticated utility propagation, but to have a sophisticated value propagation phase. We believe that the advantage of this approach is that our value propagation phase does not necessarily have to follow max-sum; it can be affixed to any approach that leaves each variable with a reduced domain of candidate values.

4.4.3 QIAAMU Formulation and Examples

In this section, we detail the steps we took to obtain a DCOP formulation of the problem of IA-QoS tradeoff in a distributed system. The solution to the DCOP specifies how each node should configure the actuators under its control such that the satisfaction of all stakeholders across all nodes is minimized, weighted by their relative importance.

4.4.3.1 Quantization and quantification

To be able to compare the desirability of different sets of QoS/IA attribute levels, we need to express how important each attribute is to each stakeholder, and how important each stakeholder is to the overall mission of the system. We therefore associate with every stakeholder s in the set of stakeholders S a weight w_s indicating the relative importance of this stakeholder. Similarly, we expresses the relative preference of stakeholder *s* to a particular attribute of a particular asset as $p_{s,a}$ where *a* refers to the attribute of an asset (e.g. Availability of database1).

Note that values of these parameters are typically going to be specific to a mission mode, a particular time interval within a mission. We therefore assume there is an implicit subscript *M* that indicates which set of values is currently being used. In the rest of this discussion, we address decision making for a given mission mode, i.e., under a given set of parameters.

We also express attribute levels (both delivered and required) in terms of quantized and ordered levels. This quantization facilitates knowledge elicitation from the stakeholders and its representation in our formulation. The requirement level desired by stakeholder *s* for attribute *a* is denoted by $q_{s,a}$.

Because environment/system conditions modulate the effects of actuators on attributes, we denote the level of attribute *a* that results from setting the actuators as dictated in the value assignment *x* under environment/system conditions *e* by $v^e_x(a)$. Table 5 summarizes the various quantities and symbols in our formulation.

Table 5: Parameters of our formulation

x	A set of values to all actuators
e	A set of values to all environment/system conditions
s	A stakeholder
a	An attribute of an asset
w_s	Weight of stakeholder s
$p_{s,a}$	Preference of s to attribute a
$q_{s,a}$	Requirement of s for the level of attribute a
$v_x^e(a)$	Level of attribute a from actuators setting x and environment/system conditions e

4.4.3.2 Cause-effect Networks

To be able to compare the desirability of different actuator configurations, we need to elicit from the domain expert how QoS/IA attributes depend on the actuators settings and environment/system conditions, i.e., we need to elicit the various $v^e_x(a)$ functions. However, when each attribute depends on a large number of actuators in a complex way, it becomes tedious and/or difficult for a user to specify each function as a flat table detailing how, for example, each configuration of firewall policies and choice of communication channel affects the Availability of HQ. It may be much easier to specify how Availability depends on Throughput and Reachability and how these depend on the actuators and system conditions.

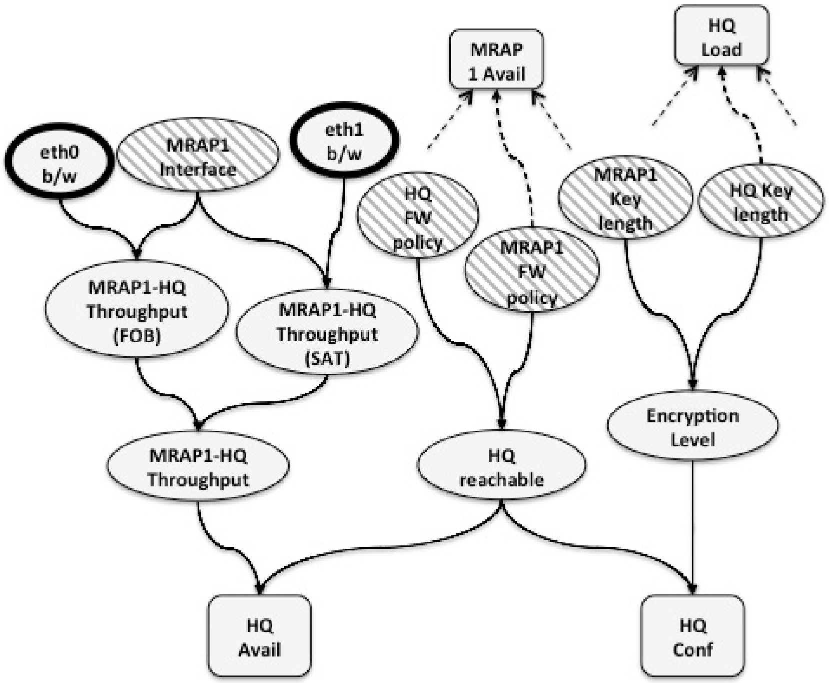


Figure 11: Part of the MRAP cause-effect network

We therefore use a representation of the set of functions $v^e_x(a)$ that is similar to Bayesian networks but with deterministic, rather than probabilistic, relations. Our *cause-effect networks* are directed graphs with nodes representing variables and edges representing cause-effect relations. Each non-root node has a table, which we call the Conditional Value Table (CVT), a deterministic version of conditional probability tables that specifies how its value depends on the values of its parents.

Figure 11 shows part of the cause-effect network for the MRAP (Mine Resistant Ambush Protected) scenario. In MRAP scenario MRAP vehicles go out to ensure a route is cleared of explosives. If they encounter suspicious activity and improvised explosive devices, they need to immediately provide visual and other information to the Headquarter (HQ). There are different ways to communicate with the information system at the HQ, the MRAP's can use line of site communication with their forward operating base (FOB), or they can use satellite communication. Line of sight and satellite communication offers different levels of QoS and security. The cause effect network shows how decisions made by one of the MRAP vehicles (MRAP1) and HQ ultimately affect the Availability and Confidentiality of the services interactions with the HQ. The decision-making nodes in this example are MRAP1 and HQ. Nodes with dark borders represent environment/system conditions (e.g. the current bandwidth of the links on eth0 and eth1 interfaces). Hashed root nodes represent actuators; the encryption key length and firewall policies used by MRAP1 and HQ, the interface MRAP1 uses (eth0 to FOB or eth1 to SAT). Intermediate nodes represent intermediate calculated values (e.g. throughput of the MRAP1-HQ connection). Dashed arrows represent other factors not shown in the figure affecting an attribute. Finally, leaves represent attributes (e.g., Confidentiality of HQ). When actuator settings and environment conditions are plugged in at the roots of the network, values of nodes at subsequent levels can be calculated from their respective CVTs, ultimately determining the values of the leaves that represent the QoS/IA levels delivered by the system under the environment conditions and chosen actuator settings.

This representation has the advantages of making explicit the structure of causes and effects among the variables. Not only is it efficient in terms of space, it is also helpful in terms of knowledge elicitation. Directly specifying how each actuator configuration affects the attributes of interest is analogous to writing out the joint probability table rather than representing it using a Bayesian Network.

4.4.3.3 DCOP formulation

We formulate the problem of finding the configuration of actuators that minimizes the satisfaction of all stakeholders as a DCOP[1]. The variables are the actuators in the system, and each variable's domain is the set of values the corresponding actuator can take. For example, the domain of the ``Encryption key length" actuator can be the set {128, 256, 512, 1024}.

The goal is to find a global assignment that minimizes the difference between the required and delivered levels of attributes for all stakeholders weighted appropriately. The objective function is the penalty incurred when these two levels do not match and can be formulated as:

$$\min_{x} \quad \sum_{s \in S} w_s \sum_{a \in A} [\, p_{s,a} * \max(0, q_{s,a} - v_x^e(a))]$$

[1] Our problem, and DCOPs in general, have some similarities to Collaborative Design Networks (CDNs) [10], but CDNs are probabilistic. Solution approaches proposed for CDNs bear strong similarity to max-sum and assume agents are arranged in a hypertree.

where the symbols are as in Table 5. The *max* operator expresses the fact that we do not care that the delivered level exceeds the required level; the best case is when these two levels match and the penalty is 0.

The above objective function can be factored into a sum over cost functions (the DCOP constraints), each of which is associated with an attribute-asset pair. The cost function of pair a is therefore

$$f_a(x) = \sum_{s \in S} w_s * p_{s,a} * \max(0, q_{s,a} - v_x^e(a))$$

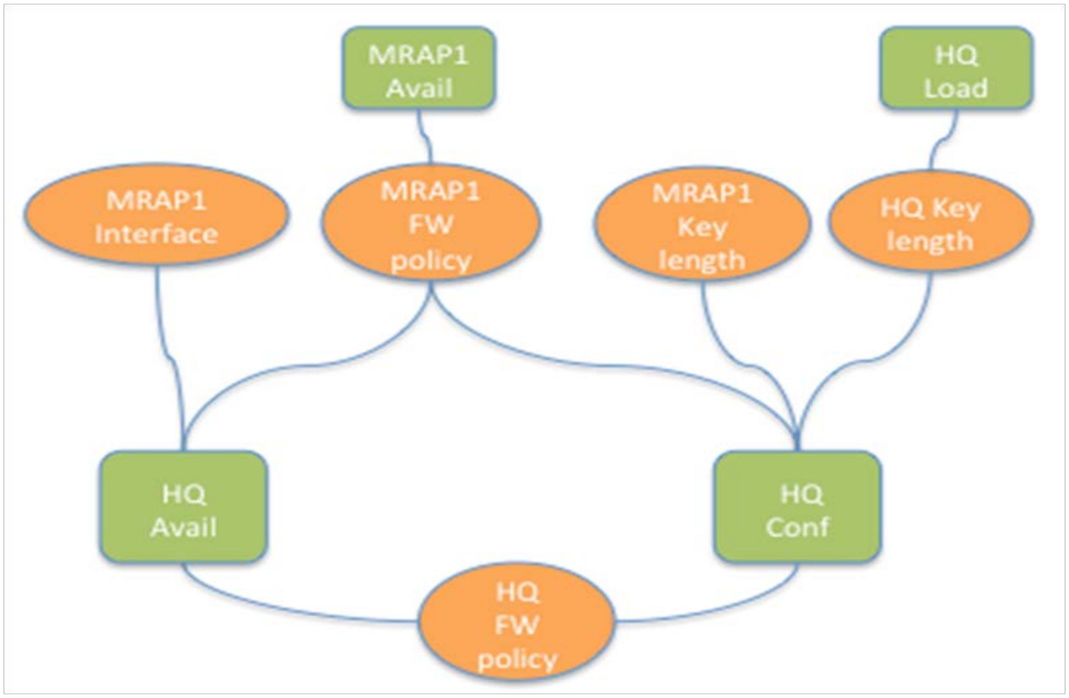


Figure 12: Part of the MRAP factor graph

Figure 12 shows the factor graph for the partial cause-effect network in Figure 11. Ovals represent variables (actuators) and rectangles represent functions (attributes). As we mentioned earlier, the graphs we tend to obtain have many cycles of various length and typically have some function nodes with a large number of neighbors. In addition, the CVTs in the cause-effect network result in HQ Availability; for example, having the same value for multiple assignments of MRAP1's interface and firewall policy and HQ's firewall policy.

As the *e* in the above formula suggests, the cost function is calculated in the context of a given set of environment/system conditions. Different sets of conditions give DCOPs that are structurally the same, but differ in their cost functions, since they differ in the functions $v^e_x(a)$. As can be seen, our cost functions are more complex than typical functions (e.g., in graph coloring). A cost function of a given attribute-asset pair is actually a cause-effect network where values of the root nodes (actuators) are plugged in and propagate through the network to yield a value for the attribute-asset pair. In the course of solving the DCOP, each function will typically be minimized many times under different assignments of the variables in its scope. This results in a large number of evaluations which, because of the complexity of our functions, can become an expensive process. To reduce this cost, we avoid re-evaluating a function from scratch; we only propagate values from the root variables that changed since the last evaluation and only re-evaluate the descendants of these roots.

4.4.4 Value Propagation and Other Extensions

Initially, we used the max-sum algorithm as described by Farinelli et. al [3] to solve our DCOP. Our choice was motivated by initial attempts at using DPOP, which proved too slow and often failed to solve our instances because it ran out of memory. As discussed earlier, the large number of neighbors a node in our factor graph typically has exacerbates DPOP's exponential message size.

In using max-sum, we soon realized that the combination of a cyclic graph and non-unique locally optimal assignments resulted in the agents reaching local assignments that sometimes have low global quality. The reason is that the local assignments reached at convergence are part of different optimal global assignments and are inconsistent with each other.

An acyclic graph with non-uniqueness can be handled by a simple value propagation phase that proceeds from the node designated as the root variable to the leaves (as suggested in [11]). After calculating its optimal value, the root node passes this value down. A function that gets a value from its parent calculates (or retrieves a cached copy of) the corresponding optimal values of its children and propagates them. A variable that gets a value from its parent passes it on to its children.

The above procedure cannot be directly applied to graphs containing cycles, since there is no longer a notion of parents and children. Even if we enforce a topological order on the nodes, a variable can potentially have multiple parents, in which case it is not clear what value it should take. Another problem we faced is that max-sum sometimes oscillates and fails to converge.

We experimented with adding small random noise to the values in the utility messages to break ties among the non-unique locally optimal partial configurations. We observed that even though ties were indeed broken, they were broken by each node locally, which still didn't address the problem of inconsistent assignments which resulted in solutions with poor global quality.

To overcome the above problems, we modified the max-sum algorithm and extended it with a value propagation phase. In this phase, utilities from the max-sum utility propagation phase are used to narrow down the domains of variables by excluding values that do not minimize z_i in Eq 1. Functions then optimize in the context of these reduced domains and suggest values to their neighbors. Because the graph is cyclic, a variable can receive multiple suggestions, so it needs to decide which suggestion to adopt. This is done using a heuristic measure of function importance. If a variable ignores a function's suggestion, the latter re-minimizes in order to obtain consistent values for the other variables in its scope.

The following sections detail this procedure.

4.4.4.1 Value Propagation

We observed that upon the termination of the utility propagation phase, it is sometimes the case that a variable has many values minimizing the function z_i in Eq 1. However, if each variable just goes ahead and chooses one of these values, the result is a globally inconsistent solution. We therefore need a value propagation phase whose goal is to assign values in a consistent manner.

The value propagation (VP) phase starts as follows:

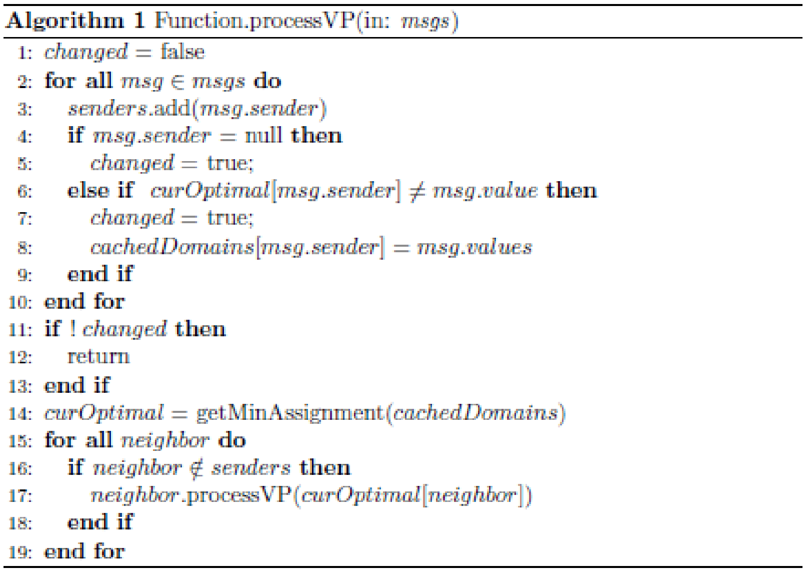
1. Each variable narrows down its domain. It performs the minimization in Eq 1 and retains only the minimizing value(s). If there is a single minimizing value, it is sent to neighboring functions in a VP message. If there are multiple values (but not the entire domain), we call this set of values *candidates*.
2. If no variable has a single minimizing value, each variable that has candidates sends VP messages to its neighbors with *null* in the sender field to signify that this variable is not suggesting any values, but just wants neighboring functions to do their minimizations using the variable's reduced domain.
3. If no variable has candidates, the *top* function is selected (more on that later) and sent an empty VP message.

In the following, we detail how each kind of node processes VP messages.

4.4.4.1.1 How a Function Processes VP Messages

A simple way of processing VP messages is for a function f_i to calculate an assignment of the variables in its scope $\boldsymbol{x}_i^*$ that minimizes its cost, with the minimization taking place over the reduced, rather than the entire, domains of variables. Each reduced domain can be a single value or a set of candidates. The function would then send out VP messages to its neighbors telling them to assign themselves according to the values in $\boldsymbol{x}_i^*$.

The problem with the above scheme is that in the presence of cycles, a variable may receive multiple conflicting assignments from its neighbors. We therefore consider a VP message from a function to a variable to be a suggestion rather than an enforcement event and allow a variable to *decline* a suggested value if it has previously adopted a different value suggested by a more important function. A variable declines a suggested value by sending a VP message back to the sender containing the other value it has taken on.



Algorithm 1 Function.processVP(in: *msgs*)

```
$changed$ = false
for all $msg \in msgs$ do
    $senders$.add($msg.sender$)
    if $msg.sender$ = null then
        $changed$ = true;
    else if $curOptimal[msg.sender] \neq msg.value$ then
        $changed$ = true;
        $cachedDomains[msg.sender] = msg.values$
    end if
end for
if ! $changed$ then
    return
end if
$curOptimal$ = getMinAssignment($cachedDomains$)
for all $neighbor$ do
    if $neighbor \notin senders$ then
        $neighbor$.processVP($curOptimal[neighbor]$)
    end if
end for
```

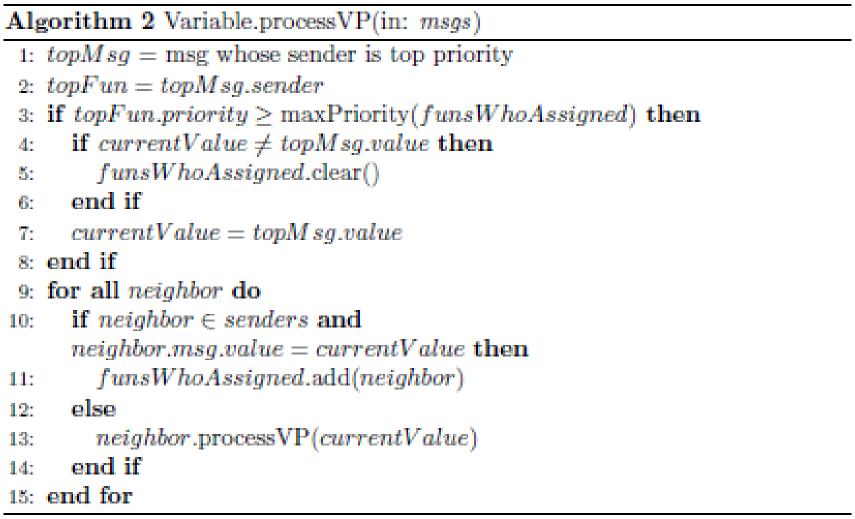
Algorithm 1 shows how a function processes a batch of VP messages. Because a function can receive multiple batches of VP messages, it can end up calculating $\mathbf{x}_i^*$ multiple times. However, some of these minimizations may be unnecessary if the messages will not change the function's opinion of what the values of its neighbors should be. Unnecessary computations can be avoided if, during the VP phase, a function keeps track of the optimal assignment (`curOptimal`) from the most recent minimization. The minimization only needs to be redone if 1) the sender variable is telling the function it has candidates, by setting the `sender` field to *null* (Line 4 or 2) one of the messages conflicts with the values in `curOptimal`, in which case the sender variable is telling the function it has declined the function's previous suggestion.

If a minimization is needed, it is followed by the function suggesting a value to each neighbor not in `senders`. If a variable was merely declaring that it has candidates, it will not be in `senders` (because the *sender* field in its message was null) and so will receive the suggestion. A neighbor that did send a message does not need to receive a suggestion because the fact that it did so means that it declined a previous value; i.e., it already reduced its domain to one value, which is what the function used in the minimization.

Another aspect of how functions process VP messages is the caching of the domains of certain neighboring variables, the reason for which will become clear after the operation of variable nodes is detailed.

4.4.4.1.2 How a Variable Processes VP Messages

Because a variable that adopts a value may later need to decline it, each variable keeps a list, `funsWhoAssigned`, of the functions that assigned it to its current value so that it can notify them if it later declines. Algorithm 2 shows how a variable processes VP messages. It starts by determining `topMsg`, the message whose sender has top priority (details below). If this sender has higher priority than any function in `funsWhoAssigned`, the variable adopts the value in `topMsg`, and if it is different from the current one, the list is cleared. The variable then sends VP messages containing its current value to a neighbor if (a) it did not receive a message from this neighbor in this round (thus propagating the value to it) or (b) the neighbor sent a message that conflicts with the new value (thus declining the value suggested in neighbor's previous message). The list `funsWhoAssigned` is re-built in the process, in case the variable needs to decline in the future.



Algorithm 2 Variable.processVP(in: $msgs$)

```
$topMsg$ = msg whose sender is top priority
$topFun = topMsg.sender$
if $topFun.priority \geq$ maxPriority($funsWhoAssigned$) then
    if $currentValue \neq topMsg.value$ then
        $funsWhoAssigned$.clear()
    end if
    $currentValue = topMsg.value$
end if
for all $neighbor$ do
    if $neighbor \in senders$ and
    $neighbor.msg.value = currentValue$ then
        $funsWhoAssigned$.add($neighbor$)
    else
        $neighbor$.processVP($currentValue$)
    end if
end for
```

4.4.4.1.3 Caching Domains

A function node keeps track of cached variable domains in `cachedDomains`, which is updated on Line 8 in Algorithm 1. Also, in `getMinAssignment`, if the function encounters a variable for which is it does not have a cached domain, it queries the variable for its domain, uses it for minimization, and updates `cachedDomains`. In the following, we give an example of why we need caching.

Consider the situation shown in Figure 13 where f_1 is connected to v_1 and v_2, each of which is connected to other functions as well. Assume each variable narrowed down its domain to the candidate set $\{a, b, c\}$. Suppose that at time 1, f_1 sent value a to v_1 and b to v_2. On receiving these messages, both variables set their domains to the values in the messages. In the figure, value messages labeled with Time:Value are shown as annotated arrows.

Suppose that at a later time, v_1 declined (because it received a suggestion, c, from a function with higher priority) and sent back a VP message to f_1 with its new value. If f_1 re-minimizes, it will do so with the domain of v_2 restricted to b. Instead, what f_1 should use is the original set of candidates of v_2, if available, that potentially has a wider set of choices rather than just b. The key point here is that v_2 reduced its domain as a result of a message from f_1, so f_1 should be able to “undo” this reduction. f_1's cached version of v_2's domain is $\{a, b, c\}$, which indeed allows it to explore all these options in re-minimizing in the context of $v_1 = c$.

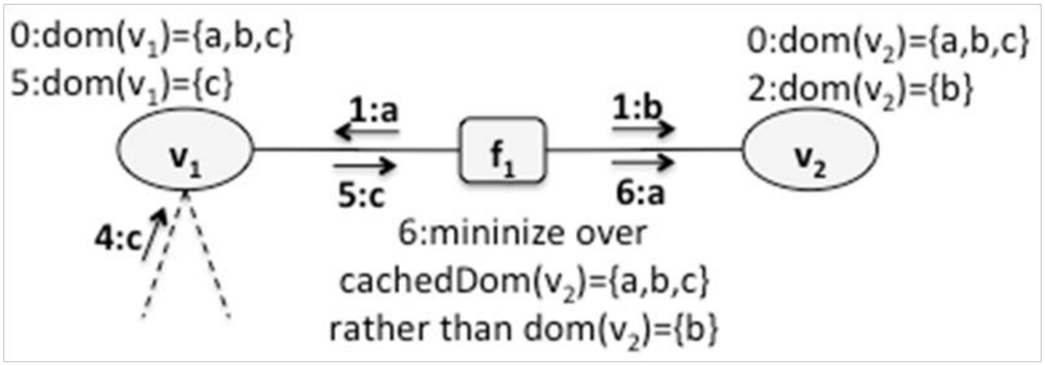


Figure 13: Illustrating the importance of Caching Domains

It is important to note that if v_2 had taken on a value suggested by a function with higher priority than f_1, v_2 would have sent a VP messages to f_1 and in processing it, f_1 would have refreshed its cache to reflect v_2's new domain and re-minimized in this new context. So overall, caching results in the desirable behavior whereby f_1's re-minimization is done in the context of values chosen by higher priority functions, if they exist, or a broader set of options, if they do not.

4.4.4.2 Function Importance and Cycle Detection

In the previous sub-section, we mentioned the *top* function on two occasions: (1) we wanted to send an empty VP message to the top function to get the VP phase started in the absence of any candidates and (2) when a variable was deciding which suggested value to take on when it receives multiple VP messages. There should therefore be a way of comparing the **relative importance** of function nodes.

Our initial heuristic was to determine importance based on the number of neighbors a function has (its degree), with the rationale that it is easier for a function with a smaller scope to optimize in the context of values suggested by a function with a larger scope, rather than the other way around. However, we realized that regardless of scope size, it is a function's contribution to the overall solution cost that matters. We therefore calculate the weight of function f_a related to attribute *a* as follows:

$$w(f_a) = \sum_{s \in S} w_s * p_{s,a} \qquad (2)$$

This equation (Eq 2) reflects the importance of the stakeholders who care about attribute *a* and how much they care about it. In our experiments, we only report results using this heuristic, since it performed consistently better than the degree-based one.

In terms of *when* messages are processed, we implemented a schedule where at the beginning of each round, an agent processes all the messages that were sent to it in the previous round in a batch. This is in contrast to an agent processing each incoming message individually and sending a set of outgoing messages in response.

In our implementation of max-sum, a node in the factor graph ignores a message from a sender if it is the same as the last message from this sender. In addition, each variable node keeps a history of states. A state is the set of most recent messages, one from each neighbor, and is updated after each batch of messages is processed. A node uses its history to **detect cycles**: situations where it is revisiting the same set of states, in which case the node ignores incoming messages and therefore produces no outgoing messages. The algorithm converges if no nodes have outgoing messages.

We terminate the utility propagation phase upon convergence or reaching a pre-set number of iterations. We found that cycle detection can slightly reduce run-time and the number of iterations. Predictably, it has no effect on solution quality. We used cycle detection for all the results we report for our implementation.

4.4.4.3 An Example

We now present a simple example that highlights how value propagation happens in the function and variable nodes. Consider a factor graph with 5 variables and 4 functions as shown in Figure 14. For simplicity, all variables are binary. Assume that at the end of the utility propagation phase, the utilities accumulated by variables v_3 and v_4 resulted in each having a single value that minimizes (Eq 1) for it (this value is `True` for v_3 and `False` for v_4. Assume that because of non-unique minimizers, the other 3 variables cannot narrow down their domains and do not have *candidates*. Assume also that according to the weights calculated by (Eq 2), function f_1 is more important than f_2, i.e. in other words, we assume $w(f_1) > w(f_2)$.

The iterations in the value propagation phase of our example proceed in the following steps.

- **Step 1:** because they have single minimizers, v_3 and v_4 send these values to their neighbors.
- **Step 2:**
  - f_1 gets the message from v_3, sets the cached domain for v_3 to T and calls `getMinAssignment` to calculate the optimal complimentary values for the other variables in its scope. Assume this call returns $\{v_1 = T, v_2 = F, v_3 = T\}$. These values are stored in `curOptimal` and sent out to v_1 and v_2.
  - f_2 does a similar process where it optimizes in the context of the received message. Assume its minimizing assignment is $\{v_1 = T, v_2 = T, v_4 = F, v_5 = F\}$. f_2 sends out to v_1, v_2, and v_5 their respective values. Note that f_1 and f_2 do not send out values to the variables they just received messages from.
- **Step 3:**
  - v_1 receives messages from both f_1 and f_2. Because $w(f_1) > w(f_2)$, `topFun` is set to f_1. Because this is the first VP message received by v_1, `maxPriority` returns 0 and v_1 adopts the value suggested by f_1 (Algorithm 2, line 6). Because the value sent by f_2 is the same as `currentValue` (the condition on line 8 is met), f_2 is added to the list of functions who assigned this value, but no message is sent to it.

- v_2 processes its incoming messages much like v_1 did, except that the value suggested by f_2 is not the same as that suggested by the top function f_1 (the condition on line 8 failed). v_2 therefore sends a message to f_2 containing the newly adopted value to let it know it declined f_2's suggestion.
- v_5 receives a single message and adopts the value suggested in it. It does not send any messages.

- ***Step 4:*** f_2 receives v_2's decline message. Because the value in the message is different from that which f_2 calculated for v_2 in a previous round (the condition on line 6 of Algorithm 1 is met), f_2 sets `changed` to `True` as a signal that it needs to re-optimize in light of this new message. f_2 also refreshes the cached domain of v_2 to F. The re-optimization is done in the context of the cached domains of v_1 and v_2. Assuming it results in the setting $\{v_1 = T, v_2 = F, v_4 = F, v_5 = T\}$, f_1 sends out messages to v_5. To avoid unnecessary communication, f_2 does not send messages to v_1 and v_4, since neither of them got a new value as a result of the re-optimization, and they have not declined their previous values.
- **Step 5:** v_5 receives the single message from f_2. Since the previous message in Step 2 was also from f_2, the condition in Algorithm 2 line 3 is met. Because the value in the message is different from the current value, `funsWhoAssigned` is cleared. v_5 adopts the suggested value and no further messages are sent.

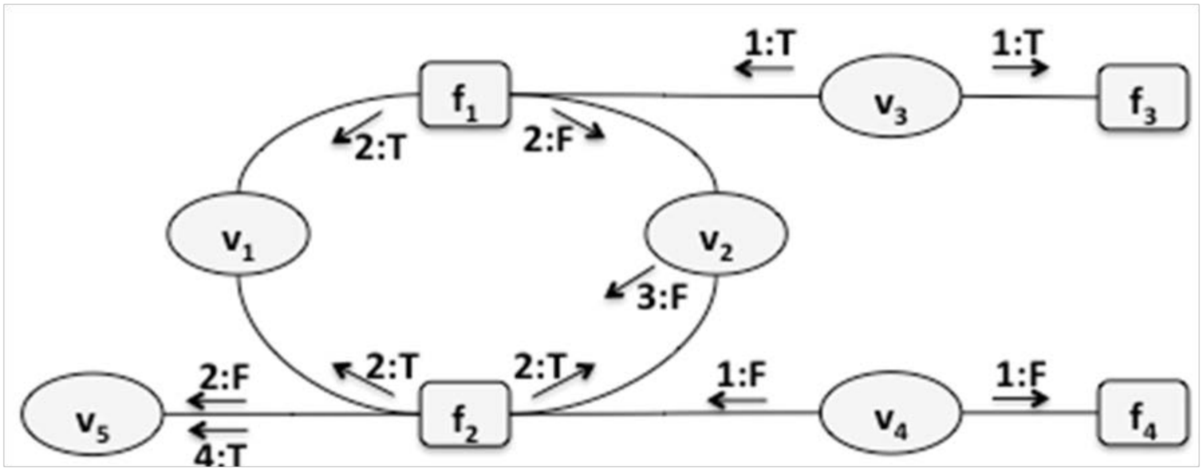


Figure 14: Value propagation example, arrows are value messages labeled with Time:Value

Let us now consider the same example without the value propagation phase. As with value propagation, v_3 and v_4 take on the values of their unique minimizers. But without value propagation, v_1 and v_2 pick values from their candidate sets independently of each other. They can potentially choose values that are inconsistent; i.e. when taken together minimize neither f_1 nor f_2. And even if they take on values that minimize the more important f_1, v_5 would take on the value F that minimizes f_2, without regard to the fact that F only minimizes f_2 if v_1 and v_2 are both set to T, which is not the case here.

To summarize, our value propagation phase has the following advantages:

1. Handling conflicting value suggestions that arise in cyclic factor graphs by allowing a variable to accept/decline values suggested by neighboring functions based on a heuristic measure of function importance.
2. Allowing a function whose suggestions are declined to re-optimize in the context of values suggested by more important functions to produce values for other variables in its scope consistent with already assigned values.
3. Avoiding unnecessary restriction of domains and allowing a sort of backtracking through the use of cached domains in re-optimizing.

4.4.5 Mitigating Distributed Implementation Challenges

Almost all the literature on max-sum, and message passing algorithms in general, focuses on theoretical and algorithmic aspects and formulating new problems as DCOPs. Implementation challenges involved in actually deploying these algorithms on multiple machines have largely been ignored (a prototype deployment is briefly described in [12]). Nodes in the factor graph are assumed to be on the same machine; a node starts out knowing its neighbors and their domain (if they are variables) and can immediately start sending messages. This simple setup is no longer `simple' when the factor graph is partitioned across machines and nodes need to discover their neighbors and communicate over the network.

In this section, we describe the challenges we faced in deploying max-sum and our value propagation (VP) phase in an actual distributed system and how we addressed them.

4.4.5.1 Message Passing Nodes

The max-sum algorithm involves passing utility (and in an optional value propagation phase, value) messages among message passing nodes (MPNs) that can be either functions or variables. When applying max-sum in QIAAMU, each host in a distributed system can have multiple actuators (variable MPNs) and attributes (function MPNs). Max-sum, however, does not distinguish between sending a message to a node on the same host and sending to a node on a remote host; it operates on a factor graph where vertices are functions and variable nodes and edges connect a function with the variables in its scope.

In order to make the local/remote distinction transparent to a MPN, we devised the class hierarchy shown in Figure 15. Functions, remote and local, implement the Function interface. Similarly, variables are also local or remote. All local nodes extend the MPN class, while all remote ones extend RemoteMPN. The Variable and Function classes implement the MPNIfc interface. This implies that all types of nodes implement the MPN interface, and then depending on whether they are local or remote, they inherit the functionality of the MPN or RemoteMPN class respectively. The advantage of this hierarchy is that a function sends messages to all its variable neighbors in the same way, regardless of where they reside.

The RemoteMPN abstract class has a reference to a service running on the remote machine. An object of type RemoteVar/Fun is initialized with the name of the remote host and a connection is established that is later used to relay messages to that host.

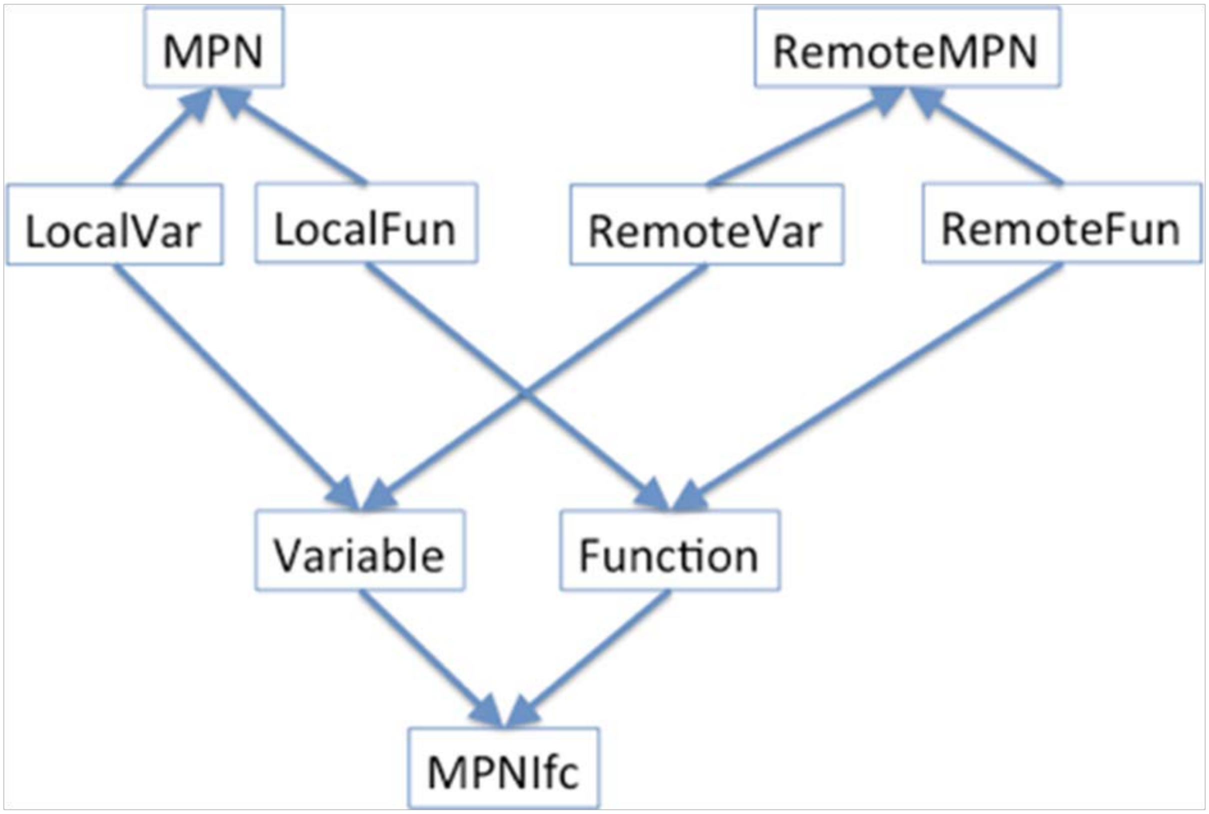


Figure 15: Message passing nodes class hierarchy

4.4.5.2 Types of Messages

There are two main types of messages that MPNs exchange:

- Max-sum messages: these are the variable-to-function and function-to-variable messages dictated by the max-sum algorithm. Utility messages encode utility tables exchanged during the utility propagation phase, and value messages are value suggestions exchanged during the value propagation phase. Max-sum messages are routed through an Orchestrator.
- Administrative messages: these are not part of max-sum itself, but are necessary for the execution of the algorithm nonetheless. Administrative messages are used during the setup phase (details below) to, for example, connect MPNs on different hosts (add neighbor messages). During the execution of max-sum, administrative messages are used to retrieve a variable's domain or ask it to narrow down that domain. Administrative messages are method calls on MPN objects. As will be detailed later, a RemoteMPN relays (some of) these calls to the host on which the actual MPN resides.

4.4.5.3 The Need for an Orchestrator

Although max-sum and VP are distributed algorithms, we believe their implementation can benefit from a small measure of centralization. We introduce an entity, the *Orchestrator* that orchestrates the QAgents and facilitates the operation of max-sum and VP. It is responsible for issues like convergence detection and preventing the system from getting into concurrent tradeoffs. Implementing distributed convergence detection for an asynchronous distributed algorithm is nontrivial [13], especially in our case where communication over a network introduces potentially highly variable delays. It also requires more communication than necessary with an Orchestrator. The same is true for the task of keeping all nodes in the same max-sum/VP iteration.

It is important to note that the Orchestrator does not generate or understand max-sum or VP messages; neither does it collect domain information (e.g. preferences) or make decisions about actuators. The Orchestrator is solely concerned with the following administrative tasks:

- Making sure all the nodes are up and managing the factor graph building process (details below).
- Managing the rounds of max-sum and VP by keeping nodes in lockstep; nodes send their outgoing messages to the Orchestrator which forwards them to their destinations and declares a new round when it receives all messages from the previous round.
- Initiating VP when the number of max-sum rounds reaches a threshold or there are no outgoing messages (convergence).
- Initiating or rejecting a tradeoff based on the outcome of running pre-tradeoff heuristics (details below).
- Ensuring max-sum and VP only start when all QAgents are in the same mission epoch.

The Orchestrator is implemented as a web service and can run on any node. Each QAgent is also implemented as a web service that manages assessment and actuation. In addition, each QAgent has a factor graph web service (FGWS) that performs max-sum related computations and message-passing for this agent. Configuration files list the address of the Orchestrator and each QAgent. We currently do not handle Orchestrator failure.

4.4.5.4 Initialization and Setup

Each QAgent reads its XML configuration, creates its part of the cause-effect network, and derives from it its portion of the factor graph, which consists of the function nodes corresponding to attributes defined in its XML, variable nodes corresponding to local actuators, and remote variable nodes corresponding to remote actuators that are connected to the local functions. Figure 16 shows parts of cause-effect networks that reside on three hosts.

The following steps describe the setup protocol followed by the Orchestrator and the QAgents:

1. Each QAgent registers with the Orchestrator which creates a connection to the QAgent's machine that will henceforth be used to communicate with it.
2. When all QAgents have registered, the Orchestrator sends them a message to create their respective portions of the factor graphs.

3. In creating the factor graph, a QAgent *A* referencing a remote actuator controlled by QAgent *B* creates a special variable proxy node that will manage administrative messages from *A* to *B*. *A* tells *B* to add it as a neighbor, which results in *B* creating a proxy function node to handle messages from *B* to *A* (details below).
4. After creating the factor graph, a QAgent knows its local variables/functions, remote actuators and which QAgents reference its local variables. Each QAgent registers its local nodes with the Orchestrator, which will use them to forward messages to the correct machines. When all QAgents have registered, the setup is complete and max-sum can start.

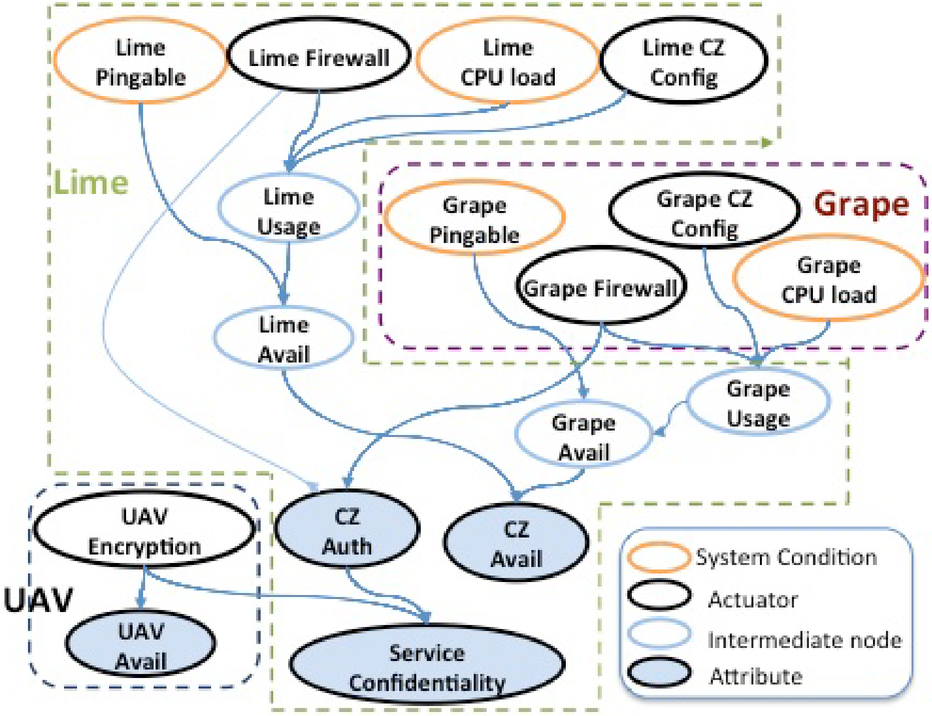


Figure 16: Partial cause-effect network distributed on *lime, grape* and *UAV* which these nodes compile into a distributed factor graph

4.4.5.5 Connecting the Factor Graph

In step 3 above, a FGWS (Factor Graph Web Service) whose blackboard contains a reference to a remote variable connects to the FGWS where that variable resides. We now detail how this process takes place such that all future communication among nodes is irrespective of location. The process is illustrated in Figure 17.

Consider two FGWSs, FGWS-A and FGWS-B, whose functions have the scopes {A1,A2,A3} and {A3,B1,B2} respectively, with the A variables residing on FGWS-A and the B variables on B. FGWS-B realizes it has a references to a remote variable (Figure 17a). It creates an object of type RemoteVar for A3 (Figure 17b). FGWS-A's XML file specifies FGWS-B as the place where A3 resides. A3's RemoteVar uses this name to look up FGWS-A's address and obtain a reference to it, to which it sends an add neighbor message (Figure 17c). Upon receiving this message, FGWS-A realizes that it needs to create a stub representing function FB to which it can send max-sum messages and which is responsible for forwarding these messages to the actual FB node on FGWS-B. FGWS-A therefore creates a RemoteFun object for FB (Figure 17d). As with the RemoteVar, FB's RemoteFun looks up and obtains a reference to FGWS-B.

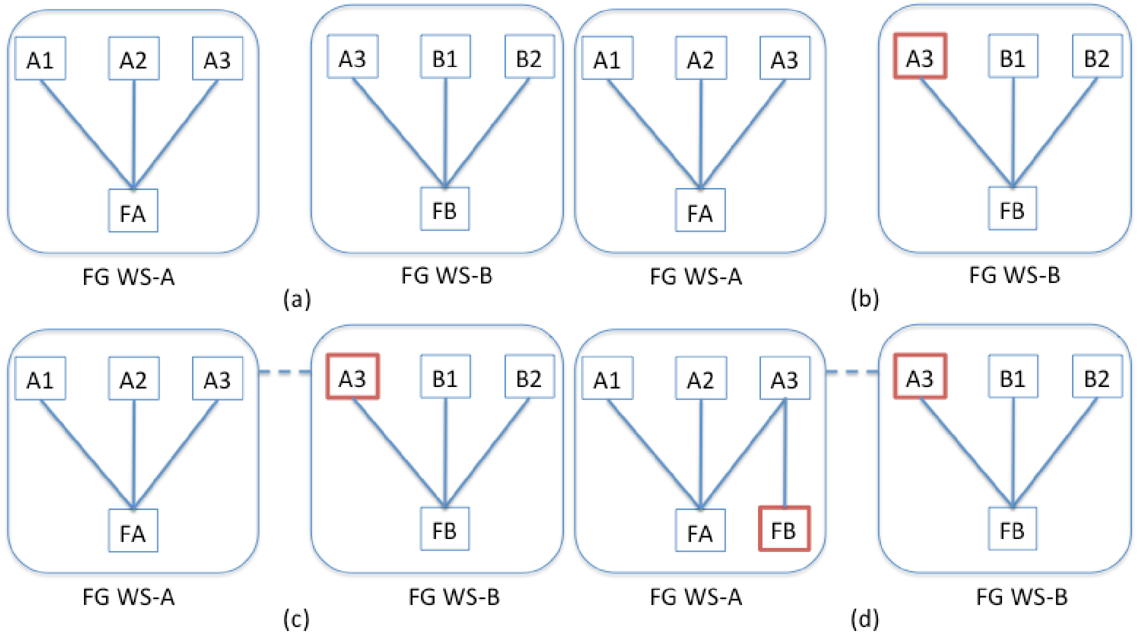


Figure 17: Steps to connect factor graph fragments

After that setup is done, administrative messages from FB to A3 proceed along the dotted line in Figure 18a, and administrative messages from A3 to FB proceed as in Figure 18b.

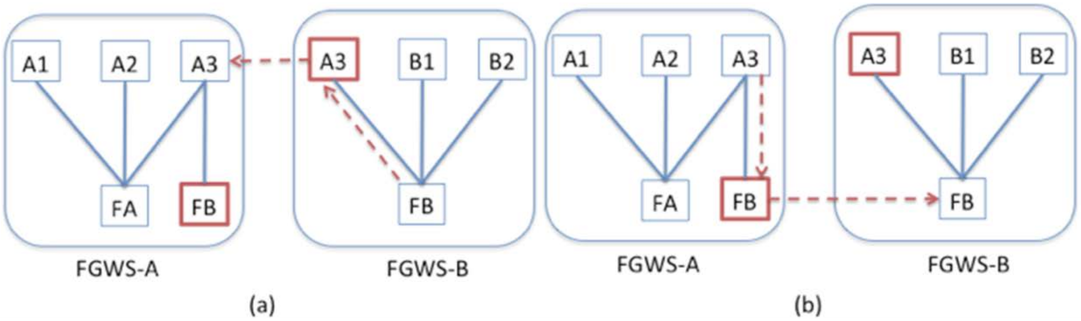


Figure 18: Administrative messages across QAgents (i.e., blackboards)

4.4.5.6 Deciding When to Initiate a Tradeoff

Because of limited CPU, bandwidth and other resources, there is typically no actuator configuration that satisfies the requirements of all stakeholders under an attack or failure. Consequently, if the Orchestrator initiates a tradeoff (invocation of max-sum followed by VP) every time some requirement is unmet, the system will be constantly running tradeoff analysis.

We devised simple pre-trade heuristics in an attempt to avoid doing a tradeoff computation if the current situation is "good enough". Upon getting a tradeoff request, the Orchestrator collects summaries from the QAgents and applies a heuristic to them. If the mission epoch has changed since the last tradeoff, or the heuristic returns true, the Orchestrator starts a tradeoff. A heuristic can judge based on whether the n most important stakeholders (by w_s) are satisfied, or the top n attribute requirements (by w_s and $p_{s,a}$) are met.

As with all aspects of our framework, the user can specify and parameterize the heuristic used in each mission epoch.

4.4.5.7 Enhancements

Enhancements made to reduce the run time of the algorithm include: reusing proxies to remote web services whenever possible, batching messages exchanged between the Orchestrator and the different factor graphs (a batch per web service rather than per variable/function), and having different web services process max-sum messages in parallel. These enhancements were made to both the utility and value propagation phases.

4.4.6 Experimental Results

We conducted experiments to compare the following algorithms and variants:

- No max-sum, only the VP phase (***NoMS-VP***): We were seeing some cases where variables end up without any candidates after the utility propagation phase, so we wanted to verify that for most cases, there is value in doing max-sum utility propagation.
- Max-sum without value propagation (***MS-NoVP***)
- Max-sum with value propagation (***MS-VP***)
- Frodo's implementation of max-sum (***F-MS***)
- Frodo's implementation of DPOP (***F-DPOP***). We include this optimal algorithm [4] to provide us with benchmark solution quality.

Frodo [14] is an open source framework for distributed constraint optimization that has implementations of several DCOP algorithms.

For all algorithms, the times we report are running times on a single machine where agents process their messages in sequence.

4.4.6.1 MRAP scenario instances

We hand-crafted seven cause-effect networks to depict six scenarios based on the MRAP mission and the underlying distributed system. Each of these DCOPs represents the tradeoff problem the runtime management framework may face during a mission.

Out of each cause-effect network, we generated a family of DCOPs by setting different values for the environment/system conditions and using different preferences of stakeholders over attributes. The DCOPs from a given scenario have the same factor graph, but differ in the constraint functions because the CVTs in the cause-effect networks are different (remember that environment/system conditions modulate the effects of actions). Each scenario resulted in a few hundred to a few thousand DCOPs.

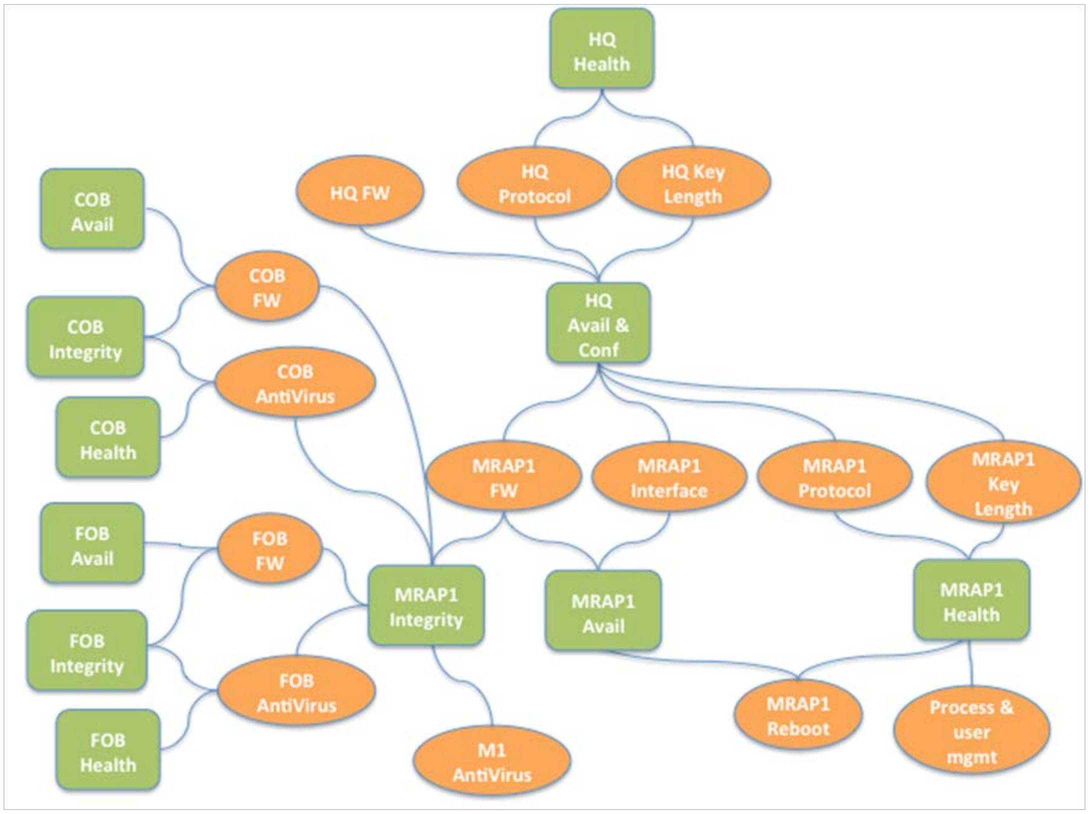


Figure 19: Factor graph of scenario 6, ovals are variables and boxes are functions

Figure 19 shows the factor graph of one of the DCOP families. Four decision makers (MRAP1, HQ, COB, FOB) decide on actuators like firewall policy (FW), antivirus policy, whether the system should reboot, communication protocol, encryption key length, process and user management (single/multiple user/process allowed at a time) and Ethernet interface to be used. These actuators affect the integrity, availability, health and confidentiality of various assets in the distributed system.

The figure illustrates a recurring feature in DCOPs from QIAAMU scenarios, namely functions with large arities. For example, the function "*HQ Avail & Conf*", which assesses the availability and confidentiality of headquarters, has 7 variables in its scope. We merged nodes that have the same set of neighbors. This has the advantage of reducing the number of cycles, without incurring the usual penalty associated with merging (an increase in a node's degree) because we only merge nodes that have the same neighbors. For variables, the result is a variable whose domain is the Cartesian product of its constituents, and for functions, the result is a function whose value is the sum of its constituents.

Table 6 and Table 7 show the times and solution costs produced by the various algorithms[2]. We found Frodo's timing results surprising; DPOP takes less time than max-sum and Frodo's max-sum takes much longer than our implementation of max-sum. The first observation is probably due to DPOP exchanging far fewer messages compared to max-sum, which is the strong point of DPOP. At the same time, the weakness of DPOP (the computational effort in calculating UTIL messages) is not manifested enough to make it slower than Frodo's implementation of max-sum, which takes a long time to converge.

The second observation can be due to the elaborate class hierarchy that allows code reuse among the several algorithms implemented in Frodo.

In fact, for the smaller networks, Frodo's DPOP took longer than a brute force centralized solver we implemented.

Table 6: Time (in seconds) and solution cost (penalty) on MRAP scenarios

	Scen 1 |V|=13 |F|=7		Scen 2 |V|=13 |F|=9		Scen 3 |V|=8 |F|=5	
	Time	Cost	Time	Cost	Time	Cost
F-DPOP	174	141	176	174	131	343
F-MS	258	184	286	225	166	395
NoMS-VP	6	182	6	254	3.2	398
MS-NoVP	46	145	47	178	15.5	355
MS-VP	49	141	49	174	18.2	349

Table 7: Time (in seconds) and solution cost on MRAP scenarios (cont.)

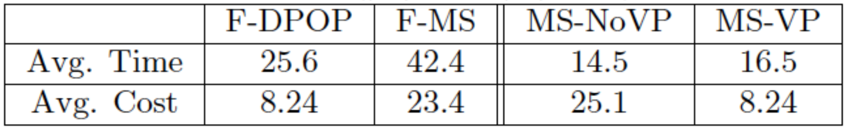
	Scen 4 |V|=11 |F|=6		Scen 5 |V|=10 |F|=5		Scen 6 |V|=15 |F|=12	
	Time	Cost	Time	Cost	Time	Cost
F-DPOP	48	23	47	42	842	124
F-MS	70	33	57	54	1194	152
NoMS-VP	2.1	30	1.7	48.6	25.3	180
MS-NoVP	9.4	23	11	42.9	204	126
MS-VP	9.8	22.7	12	42.6	212	124

[2] We report time per DCOP family, rather than per instance. Longer time is reported for a family with fewer nodes like Scen 3 because this family contains more DCOP instances.

As for the quality obtained by our max-sum implementation (without VP) compared to the Frodo implementation, it is not clear to us why our implementation gives lower cost solutions.

Table 8 shows results on a larger scenario which we call `even split'. The size of this scenario prohibits running on every DCOP representing every possible configuration of environment/system conditions, so the results shown are averages over 2000 randomly chosen configurations. Max-sum on its own (whether our implementation or Frodo's) results in solutions with much higher costs than DPOP's optimal. Using value propagation, however, is able to reach the optimal cost without significantly increasing run time compared to max-sum only, and in two thirds of the time needed by DPOP.

Table 8: Average time (in milliseconds) and solution costs on the "even split" scenario (|V| = 24; |F| = 21)



	F-DPOP	F-MS	MS-NoVP	MS-VP
Avg. Time	25.6	42.4	14.5	16.5
Avg. Cost	8.24	23.4	25.1	8.24

The thing to note about this scenario's factor graph is that it is cyclic and contains a function that operates on four binary variables, incurring a high penalty if its arguments are not evenly split (any two variables being True and the others False). This requirement arises out of the particular QoS scenario we were dealing with, which requires load balancing on a pair of servers and user preferences are such that this load balancing has a high importance. As we demonstrated earlier with examples, max-sum on its own usually results in uncoordinated choices of values in situations with cycles and multiple local optima (any combination of two True and two False values satisfy the even split function). Our value propagation phase was able to achieve the even split and avoid the high penalty.

4.4.6.2 Randomly-generated instances

Hand generating mission scenarios (something that has to be done when real system models and stakeholder requirements are fed into the QoS/IA runtime management framework) is a time consuming and tedious process, and the instances we manually generate can only get so big. We therefore resorted to randomly generating problem instances for extensive testing of our algorithm.

Initially, we used Frodo's random generator to generate binary-constrained Max-DisCSP instances, but these lacked an important characteristic that we have in distributed QoS/IA management scenarios, which is the presence of some functions with a large scope. We therefore developed our own generator and transformed its output to XCSP format so we can still run Frodo algorithms on it for comparison.

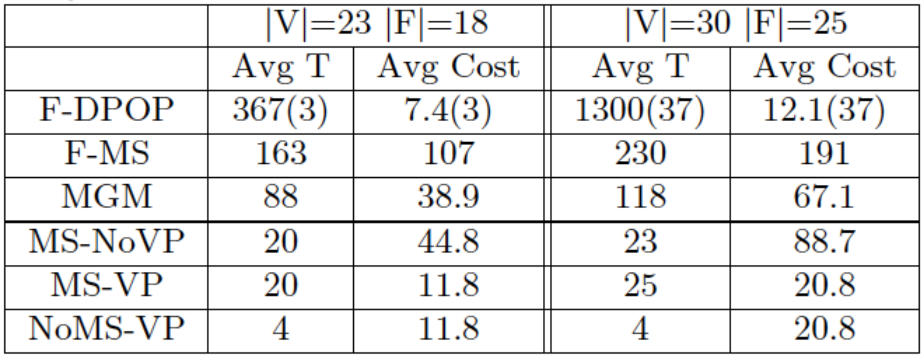
The instances we generate have loops, functions with large scopes and non-unique local minimizing assignments.

We generated 82 instances with 23 variables and 18 functions, and 82 instances with 30 variables and 25 functions. Function scopes range in size from 2 to 6. The results of running the various algorithms are shown in Table 9.

To verify that our random instances are non-trivial, we also compare to Frodo's implementation of MGM [46], a simple algorithm that should work fairly well if the instances are not too difficult.

For DPOP, the number in brackets is the number of instances it failed to solve. All other algorithms were able to solve all instances. The averages were taken over the instances an algorithm could solve.

Table 9: Average time (in milliseconds) and solution cost on random instances



	|V|=23 |F|=18		|V|=30 |F|=25	
	Avg T	Avg Cost	Avg T	Avg Cost
F-DPOP	367(3)	7.4(3)	1300(37)	12.1(37)
F-MS	163	107	230	191
MGM	88	38.9	118	67.1
MS-NoVP	20	44.8	23	88.7
MS-VP	20	11.8	25	20.8
NoMS-VP	4	11.8	4	20.8

The larger size of these random instances compared to the MRAP scenarios instances exacerbated the main problem with DPOP, namely the computation and space requirements a node needs to manipulate large incoming UTIL messages, which result from the large number of neighbors a variable has. As a result, DPOP was only able to solve a little over half the instances in the larger set. Suspecting that the message size is the problem, we tried running Memory-Bounded DPOP [6], but it still could not solve the instances that DPOP failed on.

The fact that MGM does not perform very well on these random instance assures us that they are non-trivial. As with the MRAP-based instances, having a VP phase achieves quality close to DPOP's, but an order of magnitude faster.

The most interesting result is the last row in the table which shows that running max-sum's utility propagation phase is completely useless for these instances. The cost functions are such that at the end of max-sum, the messages a variable has received do not favor any value over another. The quality of the obtained solution is solely the result of the value propagation phase which, when run without utility propagation, is one more order of magnitude faster than DPOP. We also experimented with larger instances (not shown in Table 9) with 40 variables and 160 functions. The trends were the same: DPOP was unable to solve any of the 10 instances we generated MS-VP was 1 order of magnitude faster than MGM and 2 orders faster than Frodo's max-sum. MGM produced solutions with more than twice the cost of MS-VP.

In summary, results from the MRAP scenarios and the random instances show that if used after a utility propagation phase that successfully reduces variable domains and provides each node with a good idea of the effects of its choices, our value propagation can still slightly improve solution quality without adverse effects on solution time. If performed after a utility propagation phase that results in ambiguous utilities that do not favor any subset of values, VP can still result in assignments that give solution quality comparable to that of DPOP.

4.5 Architecting for Runtime Mission-Oriented Assessment and Tradeoff Management

We envision the QIAAMU Continuous Mission-oriented Assessment and tradeoff management technology operating within and cooperating with the survivability architecture designed to defend the distributed information system that supports military scenarios. The survivability architecture provides the measurements and observations that QIAAMU needs for assessment as well as the knobs and levers (i.e., actuators) to turn in order to dynamically reconfigure the system in an attempt to offer a better tradeoff when assessment indicates that the system is not performing as expected. In this section we first explain the architecture of the runtime assessment and tradeoff management mechanism, and then show how this fits in with the overall survivability architecture of a distributed system.

4.5.1 Runtime Assessment and Tradeoff Management Architecture

The QIAAMU runtime assessment and tradeoff management consists of a distributed set of QIAAMU agents called the blackboards, an Orchestrator, and a Visualization Server. Figure 20 shows a notional organization of these components within a distributed system.

A given system will have multiple blackboards. Each blackboard has a part of the distributed system as its domain of influence. A blackboard’s domain of influence can range from a single host to a network segment.

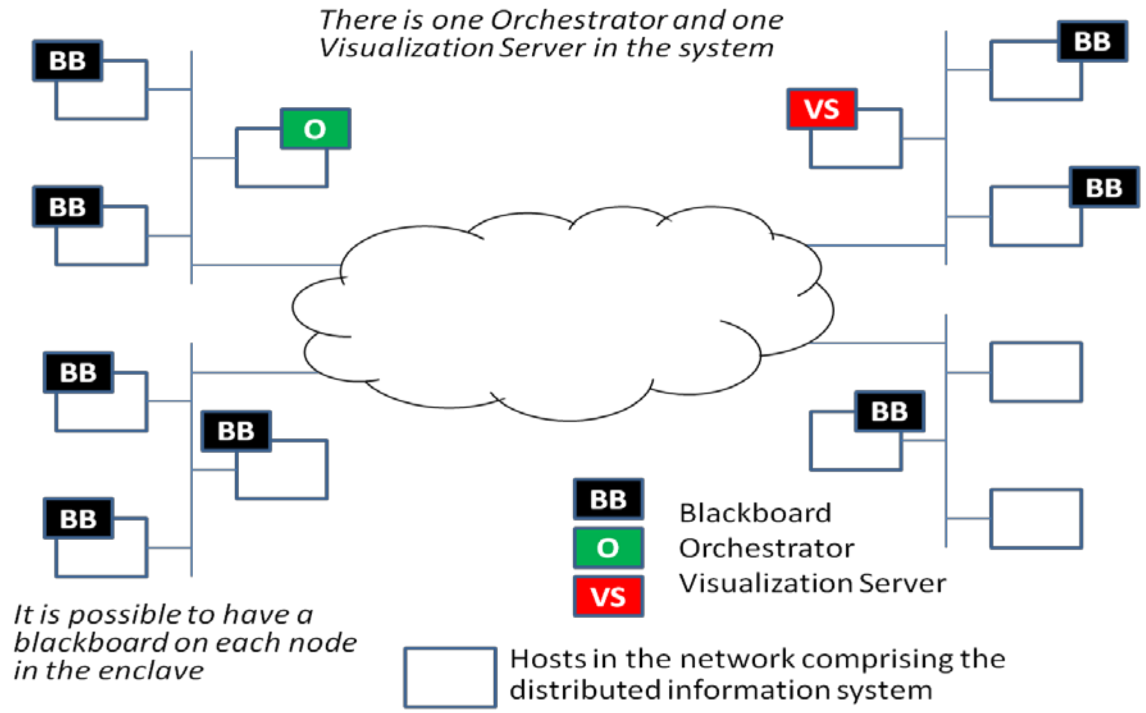

There is one Orchestrator and one Visualization Server in the system
BB
O
VS
It is possible to have a blackboard on each node in the enclave
BB Blackboard
O Orchestrator
VS Visualization Server
Hosts in the network comprising the distributed information system

Figure 20: High level organization of QIAAMU runtime assessment and tradeoff management components

In the current prototype the number and location of blackboards is determined manually by the assurance engineer. The blackboards are fairly lightweight, and can be placed on every host in the system in theory without taxing computation resources too much. However, as the number of blackboards increases, the load on the network for inter blackboard communication also increases. Another important factor to consider in the context of determining the number and location of blackboards is that the blackboards interface with the system substrate both for *sensing* and *actuating*. Enterprise System Management (ESM) databases are one of the prime sources of the raw measurements and observations that QIAAMU relies on for its runtime assessment and tradeoff computation. It is important that a blackboard is collocated with the host of the ESM databases. Similarly, blackboards should be collocated with the tunable aspects of the underlying system. The reason for this collocation recommendation is to make the interaction between a blackboard and the sensing or actuation mechanism a local (i.e., a remote) interaction.

The Orchestrator is a process that coordinates the message passing rounds of the DCOP algorithm underlying our distributed tradeoff computation. There needs to be only one Orchestrator node in the entire system, as long as it is reachable from all blackboards. The Orchestrator does maintain some state regarding the DCOP algorithm. But if it is restarted after a crash in a clean state, the only loss is efficiency. The DCOP algorithm is oblivious to the past states; the state information in the Orchestrator is needed only in coordinating the message passing rounds. If the Orchestrator state is lost, the algorithm spends more time in message passing, but works as expected.

The Visualization Server acts as the sink of log messages from all QIAAMU components, and stores partially processed information obtained from the log messages in a database. The Visualization Server includes an HTTP server that is used by QIAAMU's Javascript based runtime visualization components. Although the Visualization Server is a single point of failure, it is not a key part of the runtime assessment and tradeoff computation. If this component is lost, we will lose the runtime visualization, but the distributed computation among the blackboards will continue.

The internal organization of a blackboard is shown in Figure 21. The system condition objects represent processed measurements and observations that are used in the assessment. The actuators represent the tunable controls in the system—change in the system is effected by assigning or changing values in the actuator objects. Progress tracker is a special purpose system condition object that subscribes to mission level events and monitors elapsed time to track mission progress. There can be multiple system condition and actuator objects in a blackboard. An assessment object encapsulates the requirements for a stakeholder for a specific attribute-spatial scope combination, as well as the rules (i.e., under what system condition values) for determining when the requirements are met. A blackboard contains the assessments that are within its domain of influence. The cause and effect network captures the dependency information of how changes in the system propagate to the level of QoS and security for given spatial scopes. Like the assessment objects, the cause and effect network covers only the domain of influence of the blackboard. The tradeoff manager is the "brain" of the blackboard; it coordinates the assessment evaluations and performs the tradeoff computation. Although shown explicitly, the assessment objects, the cause and effect network and the progress trackers are essentially serving the tradeoff manager that acts as the master.

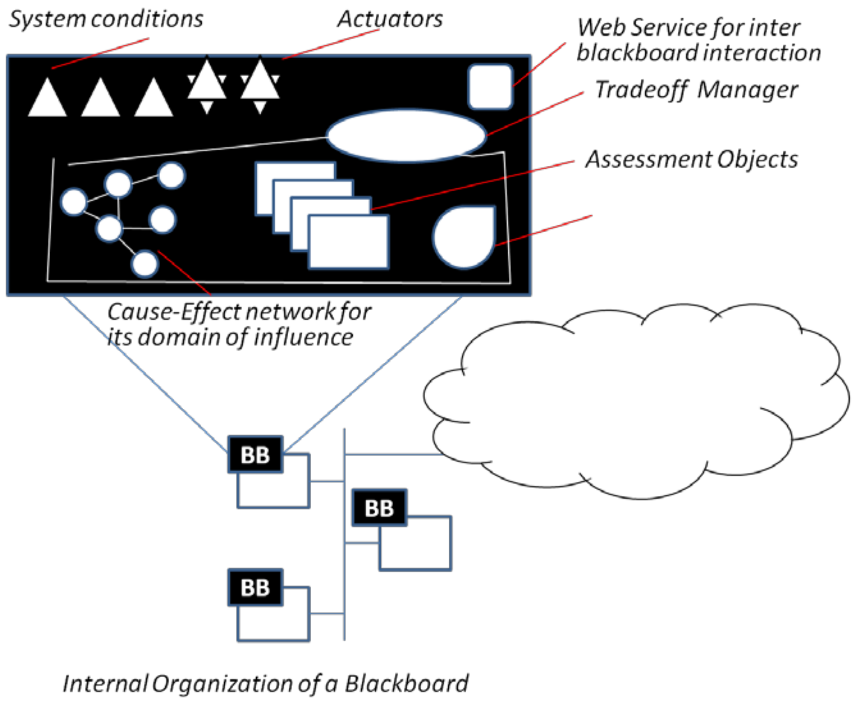

System conditions
Actuators
Web Service for inter blackboard interaction
Tradeoff Manager
Assessment Objects
Cause-Effect network for its domain of influence
BB
BB
BB
Internal Organization of a Blackboard

Figure 21: Components of a blackboard

Blackboards reach out to ESM databases, sys logs, OVAL interpreters, custom instrumentation (to individual defenses, special purpose tools) and JVMs (via JMX) for sensing and actuation.

One blackboard acts as the *primary* for each measurement or observation and actuation control (see Figure 22). Typically it is the blackboard that is collocated with the sensor source and the actuation mechanism. Other blackboards can obtain the same measurement or observation and request dynamic reconfiguring by a mechanism we call *peering*. Measurements and observations are represented within a blackboard as a system condition object. Similarly, actuator objects represent the actuation control that can be used by the blackboard to affect changes in the system. Blackboards peer with each other using a web service based protocol that allows a blackboard to declare a system condition or an actuator object as a peer to a system condition or an actuator in a remote blackboard. If the remote blackboard is the primary for the system condition or the actuator object, the local system condition or the actuator object can directly get the measurements or send requests to the primary. It is not required that peering can be done only with the primary, a system condition [or actuator] can peer with another system condition [or actuator] that itself is peered—in other words arbitrary peering network is allowed, the only requirement is that the primary must be reachable from the peering network.

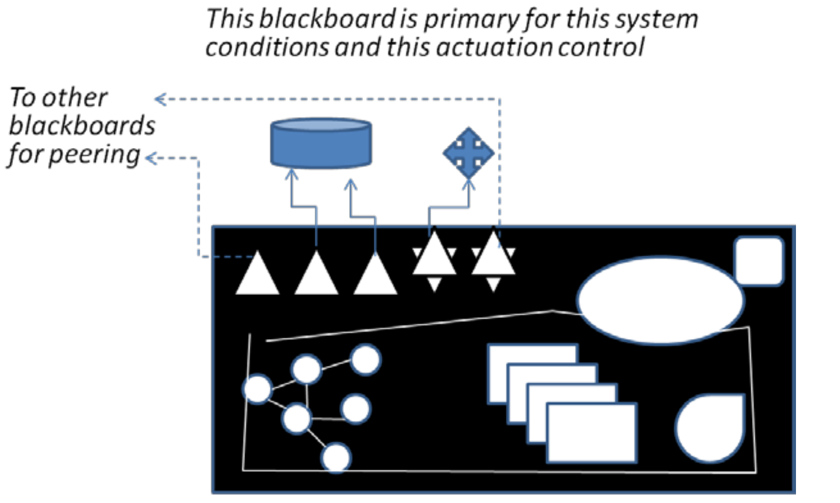

This blackboard is primary for this system conditions and this actuation control
To other blackboards for peering

Figure 22: Primary and peering relationship

Although there are a redundant number of blackboards in the system, and system condition and actuator objects can be peered, each blackboard has its own responsibility. That is, if we think of the overall assessment and tradeoff computation as a single problem, each blackboard performs its own slice; one cannot perform the role played by another. This is by design; the goal was to make the blackboards lightweight and small footprint in terms of both computation and communication- which becomes harder to achieve if one blackboard needs to take over the computation for another. However, loss of one blackboard should be expected and needs to be tolerated. This feature is not implemented in the current version, but it is designed to support partitioned operation—i.e., assessment and tradeoff computation should continue even if one or more blackboards are down or not reachable, based on the information that is available to the remaining set of blackboards. This is one of the features that will be added in the next phase of the project.

4.5.1.1 Instantiating Blackboards

As described earlier, QIAAMU blackboards must be configured to include the system conditions they are responsible for monitoring, the actuators they have the ability to control, how these elements affect (and are affected by) other elements in the local and global cause-effect network, and the requirements and preferences of the stakeholders in their domain of influence. To make it easier to instantiate blackboards, we have designed a framework where blackboards are instantiated from a configuration file, and a lot of *plumbing*- interfacing and interconnecting among the objects within the blackboard, and the outside environment is automatically taken care off. This process works as follows.

For every desired instance of a blackboard, an XML configuration file is created specifying the details of the stakeholders, system conditions, actuators and the cause and effect network that is under its purview. A Java JAR is then executed on the host where the blackboard is to operate, and will be passed the path to the appropriate configuration file. The high level structure of a blackboard configuration file is as shown below:

```
<blackboard …>
     <stakeholders>
     ...
     </stakeholders>
     <syscon .../>
     ...
     <actuator .../>
     ...
     <causeEffect>
     ...
     </causeEffect>
</blackboard>
```

The blackboard configuration, as alluded to above, consists of these four sections:

Stakeholders: Each stakeholder that has a requirement defined over an element specific to the given blackboard will be listed, given a name, and an importance. The requirements a stakeholder specifies are particular to a given mission mode, thus a series of “mode_specs” nodes are present within each stakeholder XML block. These specs specify which mode is being described, and include a node for each preference of concern. Shown below is a partial snippet from a stakeholder description of a JTAC actor. The JTAC has a requirement for the confidentiality of its data as it flows through the enterprise to be at a mid-level (specified by “M” below) and a preference value of 6 is given to this particular requirement.

```
<stakeholder name="JTAC" importance="1">
     <mode_specs name="ISR Acquisition">
          <preference
               attr="enterpriseConfJTAC"
               pref="6.0"
               req="M" />
```

The content in the stakeholder section provides the bulk of the information needed for the assessment objects.

System Conditions: Each blackboard has a set (0 or more) of available system conditions that it has access to, and that can be used to draw conclusions about the state of QoS and security in its local view. These system conditions may be gathered from system logs, database tables, querying running process state, etc. QIAAMU has a library of system condition types that it is aware of (note these are high level types, such as “relational database”, not instances specific to a particular resource, platform, or tool). Since these types are general they encompass a great many of the available sources of information. If a source exists that is not covered by the current library, the library can be extended by providing a custom Java class that implements a specific interface. The XML to configure each system condition type will be different, as the required parameters to interface with various kinds of information sources will vary (an RDBMS may require a host, port, database name, query, etc. while a log file may require a path and a regular expression, for example).

QIAAMU currently supports configuration and use of the following high level system condition types:

- Relational database backed information
- JMS exposed information
- Syslog formatted logging files
- Reachability between two nodes
- Point to point bandwidth between two nodes
- Host CPU usage
- Crumple Zone configuration state
- Firewall configuration
- Process alive state
- Process memory usage

Note that a variety of specific information not readily apparent from the above list is available to QIAAMU through the combination of existing resource monitoring tools and a generic element from the library such as a relational database, or syslog file. For example, JBoss Application Server status and the results of OVAL scans are both readily used in existing QIAAMU simulations via the set of elements above in combination with other tools available on the target systems. The below partial example depicts a system condition configuration that reads the number of failed single-packet-authorization “knocks” from a database table.

```
<syscon name="limeFailedKnocks" datatype="Long" source="DB"
     database="ossec"
     query="
       SELECT count(*)
       FROM alert a
          inner join signature s on a.rule_id = s.rule_id
          inner join `data` d on d.id = a.id
          where     s.description = ...
```

In addition to specifying the details of a local or locally available system condition, a blackboard may need to specify a reference to a remote system condition, managed by another blackboard that the local blackboard needs access to (i.e., peered system conditions). An example is shown below:

```
<syscon name="grapePingable"
     datatype="Long"
     source="peer"
     peer="grape"
     remoteName="grapePingable" />
```

Actuators: In addition to the system conditions that a blackboard can read, it may also have some number of actuators that it can engage. These represent the tunable parameters of a system that a blackboard can interact with. Similar to the generic library of system conditions, QIAAMU makes available a library of actuator types. Again these library elements know how to interact with systems at a high level (i.e. JMS, or executing a script), and are not defined for specific tools (rather they represent the interaction patterns common across many tools).

The library currently contains high level actuation elements for:

- JMX Interactions
- Script Executions
- Crumple Zone Configuration
- Firewall/Single-Packet-Authorization Configuration

Two example actuator configurations are shown below. The first is an actuator for executing a script. In this case the scripts can modify a firewall to be in one of two states: lenient or strict.

```
<actuator name="limeFWSPA"
     type="ExternalScriptActuator"
     datatype="String"
     successString="SUCCESS"
     defaultDirectory="qiaamu-scripts/firewallScripts"
     defaultExtension=".sh"
     systemCondition="IPTablesSystemCondition">
     <values initial="lenient">
          <value val="lenient" />
          <value val="strict" />
     </values>
</actuator>
```

This second actuator enables QIAAMU to interact via the Java Management Extensions (JMX) to modify an available attribute of the Java process. In this case the process is exposing the ability to set its compression level to one of three values (LOW, MED, HIGH).

```
<actuator name="jtacCompress"
     type="JMXDynamicMBeanActuator"
     datatype="String"
     shared="true"
     jmxURL="service:jmx:rmi:///jndi/rmi://localhost:9995/server
"
     user="controlRole"
     pass="crpasswd"                                       mbean-
     Name="com.bbn.qiaamu.simulation.client:type=ClientStateActu
     ator"
     attributeName="compression"                 systemCondition-
     Type="JMXSystemCondition">
     <values initial="LOW">
          <value val="LOW" />
          <value val="MED" />
          <value val="HIGH" />
     </values>
</actuator>
```

Cause Effect Network: QIAAMU must understand how the various elements available to it (system conditions, actuators) impact the requirements of stakeholders. The system conditions and actuators are, however, at too low of a level to be meaningful to a stakeholder for the purposes of defining requirements. A non-technical stakeholder, for example, will likely not wish to specify that they need to see some specific text from a log file to meet their need, or even a particular throughput on a given link. Instead their requirements are likely to be against higher level concepts such as availability, confidentiality, etc. These higher level concepts may be built from a complex combination of the lower level system condition and actuator values. A cause-effect network describes how low level conditions impact intermediate concepts and eventually map to the high level attributes of concern to stakeholders.

The example below depicts a single node from the cause effect network. A given blackboard may have many such nodes referencing other local nodes or remote nodes that are defined on other blackboards. This node represents the determination of the confidentiality of the JFO's data as it flows through the enterprise. It is built up from two parent nodes: one represents the JFO's encryption level, and another represents the crumple zone's authentication and authorization (AA) level. In this case the values are combined using a weighted scaled average, giving encryption a weight of 60% of the value, and the AA level a weight of 40%.

```
<node attribute="true" name="enterpriseConfJFO">
     <parents>
          <parent name="jfoEncryption" />
          <parent name="czAA" />
     </parents>
     <evaluator type="weightedScaledAverage">
          <contribution
               source="jfoEncryption"
               min="0"
               max="1"
               weight=".60" />
          <contribution
```

```
               source="czAA"
               type="Level"
               weight=".4" />
     </evaluator>
</node>
```

Cause-effect nodes can have their value determined in a handful of ways, including the scaled average approach shown above, but also by specifying Java code snippets as in the example below. These code snippets are compiled into Java classes at blackboard startup time expose the ful power of Java to QIAAMU cause-effect networks.

```
<node name="limeClientCtr" datatype="Integer">
     <parents>
          <parent name="casCrumpleZone" />
          <parent name="uavCrumpleZone" />
          <parent name="jtacCrumpleZone" />
          <parent name="jfoCrumpleZone" />
     </parents>
     <evaluator type="code">
          <code>
               <![CDATA[
               int ctr = 0;
               for (int i=0; i<4; i++)
               if (parents.get(i).curValue.equals("lime"))
                    ctr++;
               return ctr;
              ]]>
          </code>
     </evaluator>
</node>
```

In addition to defining the network used to allow QIAAMU to select appropriate actuation decisions, QIAAMU exposes the concept of a trigger to allow for more deterministic/predictable actions to be taken. It may be the case that if a particular condition is met, that the system should always take an action. An email could be sent out in the case of near disk resource exhaustion, for example. In the example below a log message is generated every time the firewall on Lime is set to lenient. Triggers can be set to execute only the first time their condition is met, every time their condition is met (as seen below with the "ON_EVERY" repeatability value), or only when the condition is met and was not met in the previous check of this trigger.

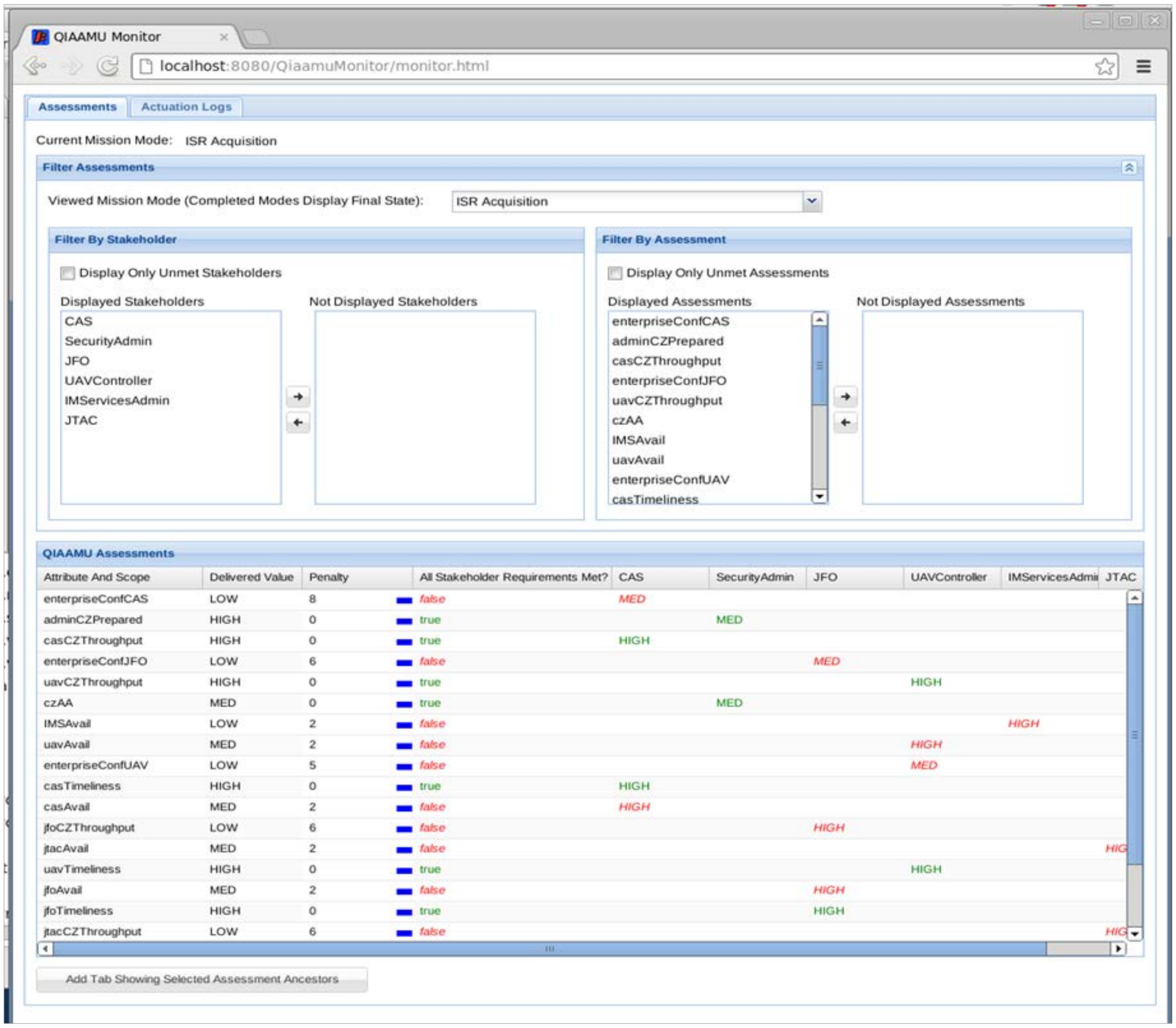
```
<node name="testTrigger" trigger="true"
     triggerType="LoggingTrigger"
     triggerRepeatability="ON_EVERY">
     <parents>
            <parent name="limeFWSPA"/>
     </parents>
     <evaluator type="code">
          <code>
             <![CDATA[
              if(limeFWSPA.toLowerCase().equals("lenient")) {
                  addState("message", "Lime firewall's current
                                    status is: " + limeFWSPA);
                     return true;
                  } else {
                     return false;
                  }
                ]]>
          </code>
     </evaluator>
</node>
```

4.5.1.2 Runtime Visualization

4.5.1.2.1 QIAAMU Visualization

The QIAAMU runtime visualization is a browser-based GUI that displays the status of the runtime assessment and tradeoff management mechanism. The web pages displayed in the browser-based GUI are implemented in Java Script and Ext JS (A Java Script framework). The web pages are served by the Visualization Server component described earlier. The server side, which is implemented in Java, interfaces with the SQL database that stores logs produced by blackboards and the Orchestrator. The monitoring GUI polls the visualization server for updated information every 10 seconds, and receives the latest data that is stored in the server when it requests it.

The GUI displays several pieces of information about the runtime state of the QIAAMU assessment and tradeoff management, including the current mission mode, the stakeholder requirements, if the stakeholder requirements are being met, details about the current system status and why the stakeholder requirements are not being met, and the history of any actuation that has taken place. The current mission mode, stakeholder requirements, and if they are being met are displayed on the first tab of the monitoring GUI (see Figure 23). The second tab holds a table of actuation log messages. Additional tabs may be opened by the users that contain a snapshot in time of the conditions pertaining to an assessment.



Current Mission Mode: ISR Acquisition

Filter Assessments

Viewed Mission Mode (Completed Modes Display Final State): ISR Acquisition

Filter By Stakeholder

□ Display Only Unmet Stakeholders

Displayed Stakeholders: CAS, SecurityAdmin, JFO, UAVController, IMServicesAdmin, JTAC

Not Displayed Stakeholders:

Filter By Assessment

□ Display Only Unmet Assessments

Displayed Assessments: enterpriseConfCAS, adminCZPrepared, casCZThroughput, enterpriseConfJFO, uavCZThroughput, czAA, IMSAvail, uavAvail, enterpriseConfUAV, casTimeliness

Not Displayed Assessments:

QIAAMU Assessments

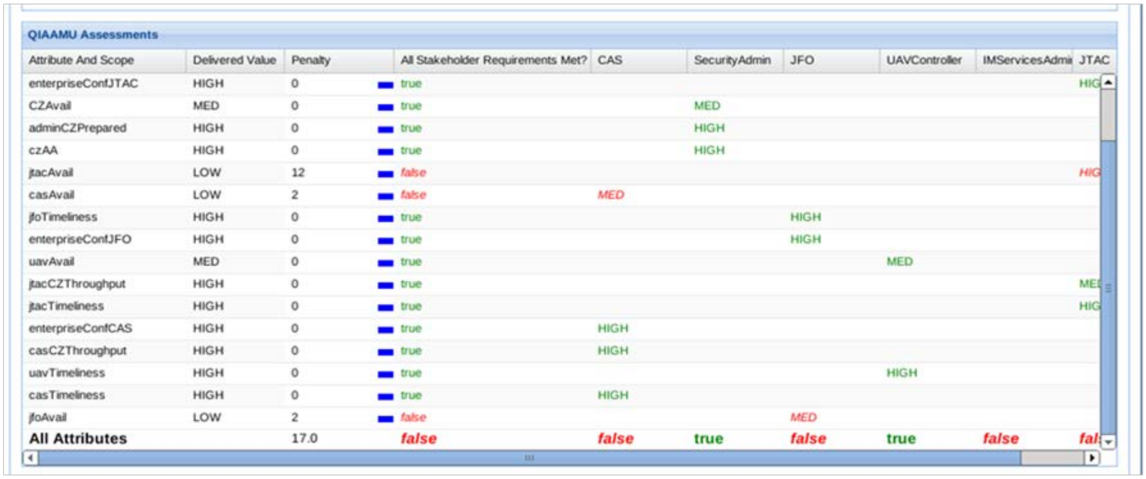
Attribute And Scope	Delivered Value	Penalty	All Stakeholder Requirements Met?	CAS	SecurityAdmin	JFO	UAVController	IMServicesAdmi	JTAC
enterpriseConfCAS	LOW	8 ▬	false	MED					
adminCZPrepared	HIGH	0 ▬	true		MED				
casCZThroughput	HIGH	0 ▬	true	HIGH					
enterpriseConfJFO	LOW	6 ▬	false			MED			
uavCZThroughput	HIGH	0 ▬	true				HIGH		
czAA	MED	0 ▬	true		MED				
IMSAvail	LOW	2 ▬	false					HIGH	
uavAvail	MED	2 ▬	false				HIGH		
enterpriseConfUAV	LOW	5 ▬	false				MED		
casTimeliness	HIGH	0 ▬	true	HIGH					
casAvail	MED	2 ▬	false	HIGH					
jfoCZThroughput	LOW	6 ▬	false			HIGH			
jtacAvail	MED	2 ▬	false						HIG
uavTimeliness	HIGH	0 ▬	true				HIGH		
jfoAvail	MED	2 ▬	false			HIGH			
jfoTimeliness	HIGH	0 ▬	true			HIGH			
jtacCZThroughput	LOW	6 ▬	false						HIG

Figure 23: Main panel for runtime visualization

Various features of this panel are explained below.

Mission modes: QIAAMU assumes that a mission consists of different epochs, and stakeholders QoS and Security requirements vary over time during the mission depending on the mode of the mission. Within one mission mode, stakeholder requirements remain the same, but may change when a new mode is entered. The GUI shows the current mission mode, and also stores the final state of a mission mode so the user has the option to look back at previous mission modes (by choosing the Viewed Mission Mode in the Filter Assessment subpanel).

Assessments: The stakeholder requirements and the attributes contributing to the requirements being met are displayed in the first tab in the assessment table (see Figure 24). This table displays which stakeholder requirements are being met and the attributes contributing to those requirements being met. The first column displays the attribute and scope. The next two columns display the value and the penalty associated with that attribute/scope. The penalty column contains both a number and an icon. The higher the penalty number, the worse the assessment is, so low penalties are desired. The icon indicates if the penalty has increased, decreased, or stayed the same since the last update. The icon will be a blue rectangle for the penalty staying the same, a red up arrow for the penalty increasing, or a green down arrow for the penalty decreasing. The fourth column indicates if all the stakeholders who care about that attribute/scope have their requirement met or not. From the fifth column on, there is one column per stakeholder. If that stakeholder has a requirement for a particular attribute/scope, their required value is displayed in the corresponding row. If the requirement is being met, the text will be the color green, and if it is not being met, the text will be the color red. The last row of the table is a summary row. This row displays the sum of all penalties in the penalty column, displays if all stakeholders have their requirements met or not, and for each of the individual stakeholders displays if that stakeholder has their requirements met or not.



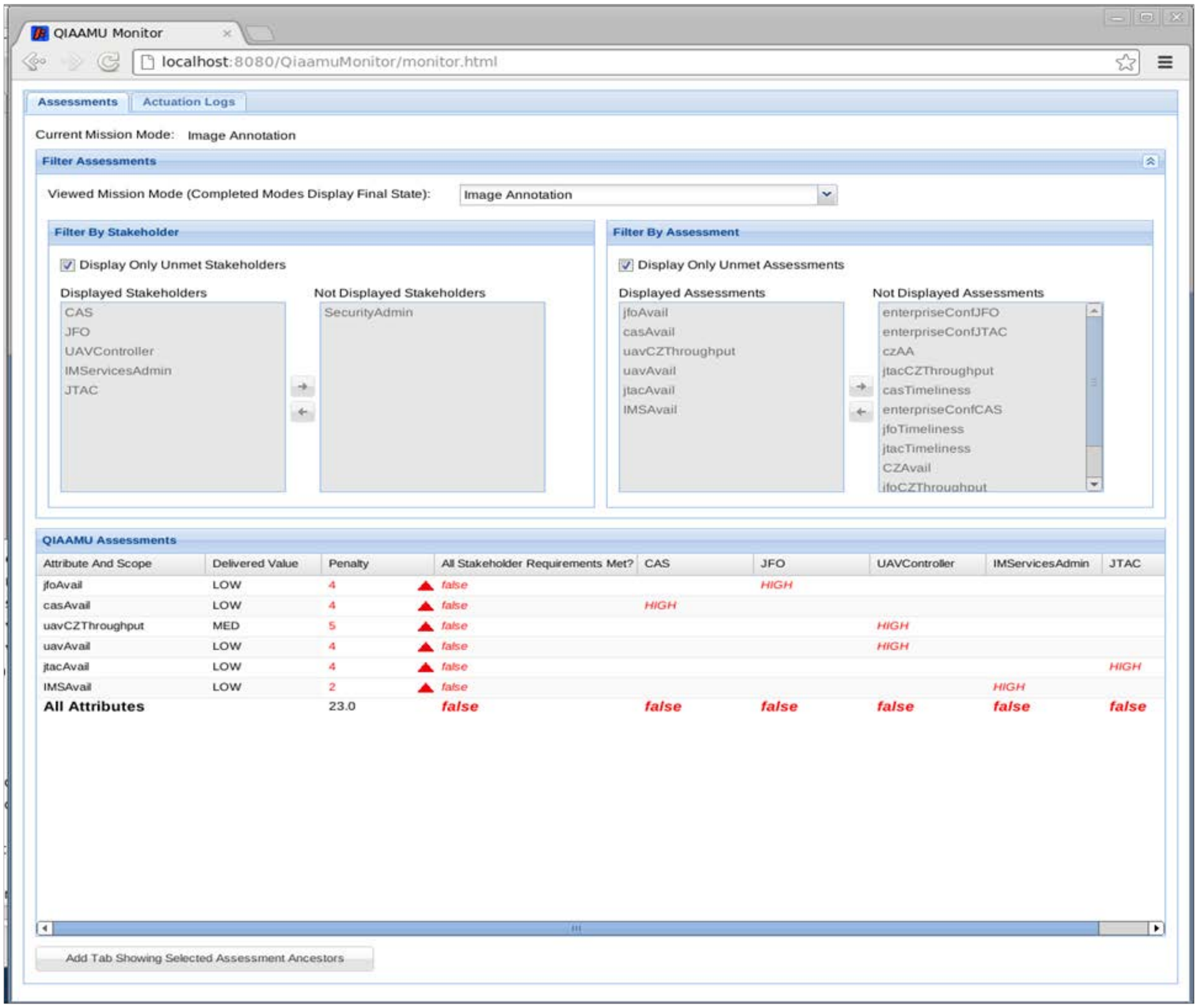
QIAAMU Assessments

Attribute And Scope	Delivered Value	Penalty	All Stakeholder Requirements Met?	CAS	SecurityAdmin	JFO	UAVController	IMServicesAdmi	JTAC
enterpriseConfJTAC	HIGH	0 ▬	true						HIG
CZAvail	MED	0 ▬	true		MED				
adminCZPrepared	HIGH	0 ▬	true		HIGH				
czAA	HIGH	0 ▬	true		HIGH				
jtacAvail	LOW	12 ▬	false						HIG
casAvail	LOW	2 ▬	false	MED					
jfoTimeliness	HIGH	0 ▬	true			HIGH			
enterpriseConfJFO	HIGH	0 ▬	true			HIGH			
uavAvail	MED	0 ▬	true				MED		
jtacCZThroughput	HIGH	0 ▬	true						ME
jtacTimeliness	HIGH	0 ▬	true						HIG
enterpriseConfCAS	HIGH	0 ▬	true	HIGH					
casCZThroughput	HIGH	0 ▬	true	HIGH					
uavTimeliness	HIGH	0 ▬	true				HIGH		
casTimeliness	HIGH	0 ▬	true	HIGH					
jfoAvail	LOW	2 ▬	false			MED			
All Attributes		17.0	***false***	***false***	**true**	***false***	**true**	***false***	***fal***

Figure 24: Assessment table with a summary row

The status displayed in the assessment table depends on which mission mode is selected for viewing. The viewed mission mode may be selected with the viewed mission mode drop down menu in the filter assessments panel. If the current mission mode is selected, the most recent status will be displayed in the assessment table. When a switch occurs to a new mission mode, this mode will automatically be selected. If a previous mission mode is selected, the final state of that mission mode will be displayed. If no changes are made using the viewed mission mode drop down, the current status will always be displayed.

Filtering (customizing) Views: The assessment table may be filtered to only show the desired results. It can be filtered either by stakeholder or assessment (see Figure 25 for an example). There is the option to view only stakeholders whose requirements are not met, and also the option to view only assessments for which some stakeholder requirements are not met. These options can be selected using check boxes in the filter panel. There are also two lists for stakeholders and assessments. One list contains the displayed items, and another the not displayed items. The stakeholders or assessments can be moved between the lists to allow the user to only view the items of importance to them. When the display only unmet checkboxes are selected, the lists are not enabled because the assessment location is determined by the assessment being met or not.



Current Mission Mode: Image Annotation

Filter Assessments

Viewed Mission Mode (Completed Modes Display Final State): Image Annotation

Filter By Stakeholder

☑ Display Only Unmet Stakeholders

Displayed Stakeholders	Not Displayed Stakeholders
CAS JFO UAVController IMServicesAdmin JTAC	SecurityAdmin

Filter By Assessment

☑ Display Only Unmet Assessments

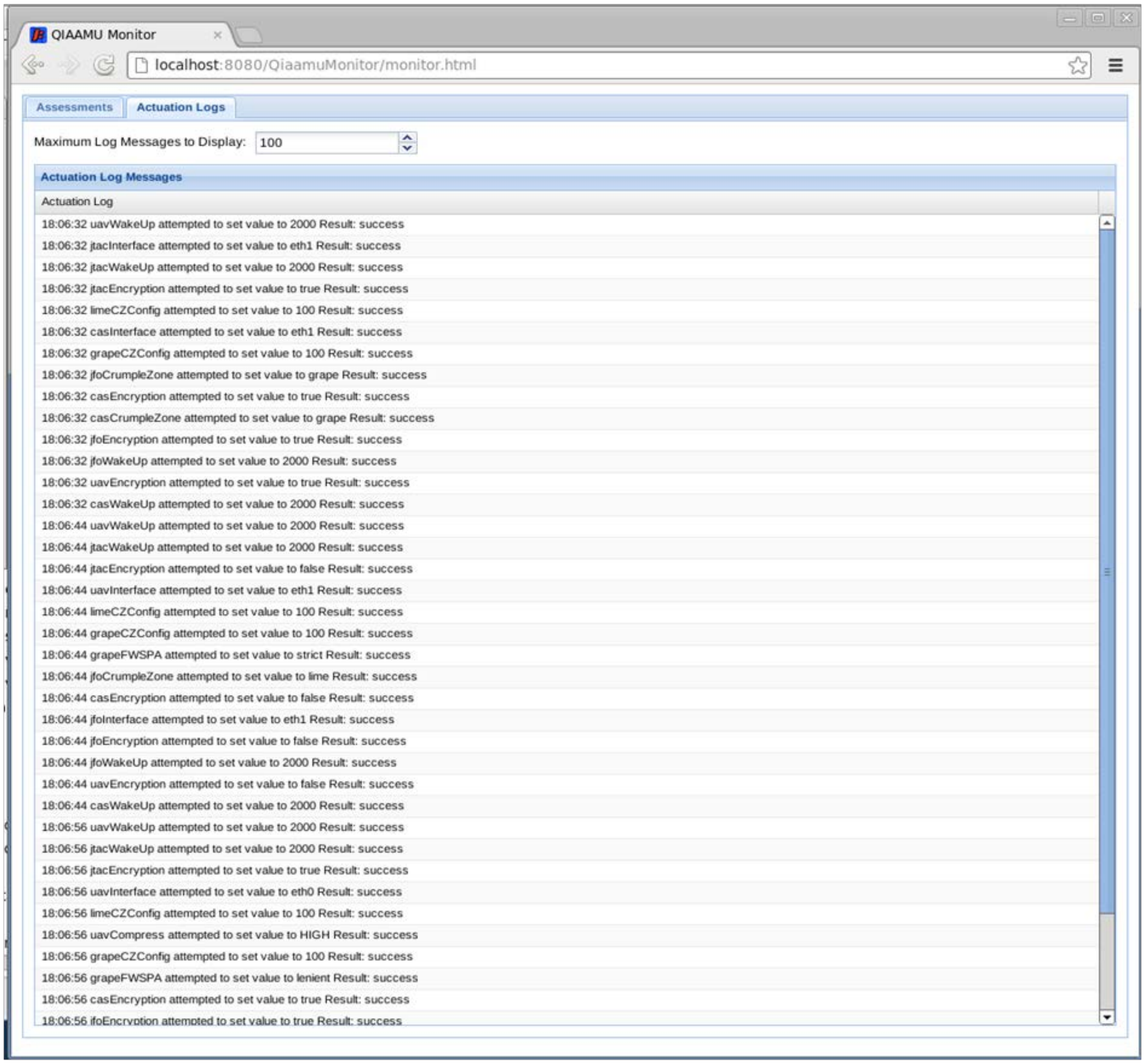
Displayed Assessments	Not Displayed Assessments
jfoAvail casAvail uavCZThroughput uavAvail jtacAvail IMSAvail	enterpriseConfJFO enterpriseConfJTAC czAA jtacCZThroughput casTimeliness enterpriseConfCAS jfoTimeliness jtacTimeliness CZAvail jfoCZThroughput

QIAAMU Assessments

Attribute And Scope	Delivered Value	Penalty	All Stakeholder Requirements Met?	CAS	JFO	UAVController	IMServicesAdmin	JTAC
jfoAvail	LOW	4	▲ false		HIGH			
casAvail	LOW	4	▲ false	HIGH				
uavCZThroughput	MED	5	▲ false			HIGH		
uavAvail	LOW	4	▲ false			HIGH		
jtacAvail	LOW	4	▲ false					HIGH
IMSAvail	LOW	2	▲ false				HIGH	
All Attributes		23.0	false	false	false	false	false	false

Figure 25: Filtered view of assessments

Actuation Logs: The second tab allows the actuation logs to be viewed (see Figure 26). If QIAAMU has decided that service could be improved by changing the value of an actuator, then an actuation will occur. This actuation will trigger a log message to be generated. The time stamp associated with this actuation and the actuation log message itself can be viewed in the list.



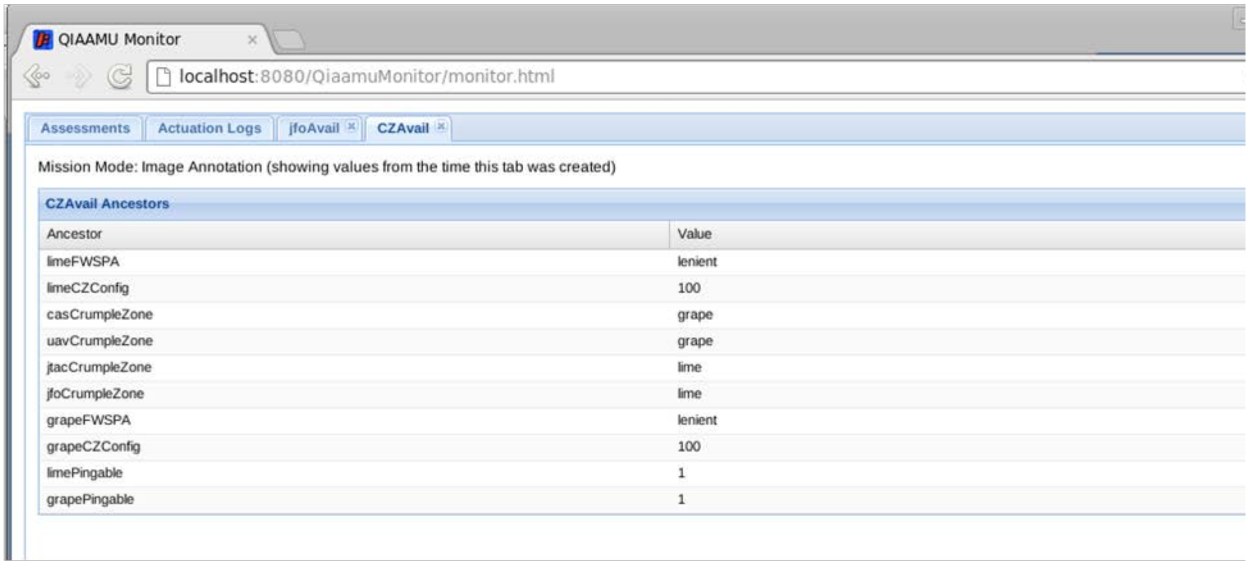
Maximum Log Messages to Display: 100

Actuation Log Messages

```
Actuation Log
18:06:32 uavWakeUp attempted to set value to 2000 Result: success
18:06:32 jtacInterface attempted to set value to eth1 Result: success
18:06:32 jtacWakeUp attempted to set value to 2000 Result: success
18:06:32 jtacEncryption attempted to set value to true Result: success
18:06:32 limeCZConfig attempted to set value to 100 Result: success
18:06:32 casInterface attempted to set value to eth1 Result: success
18:06:32 grapeCZConfig attempted to set value to 100 Result: success
18:06:32 jfoCrumpleZone attempted to set value to grape Result: success
18:06:32 casEncryption attempted to set value to true Result: success
18:06:32 casCrumpleZone attempted to set value to grape Result: success
18:06:32 jfoEncryption attempted to set value to true Result: success
18:06:32 jfoWakeUp attempted to set value to 2000 Result: success
18:06:32 uavEncryption attempted to set value to true Result: success
18:06:32 casWakeUp attempted to set value to 2000 Result: success
18:06:44 uavWakeUp attempted to set value to 2000 Result: success
18:06:44 jtacWakeUp attempted to set value to 2000 Result: success
18:06:44 jtacEncryption attempted to set value to false Result: success
18:06:44 uavInterface attempted to set value to eth1 Result: success
18:06:44 limeCZConfig attempted to set value to 100 Result: success
18:06:44 grapeCZConfig attempted to set value to 100 Result: success
18:06:44 grapeFWSPA attempted to set value to strict Result: success
18:06:44 jfoCrumpleZone attempted to set value to lime Result: success
18:06:44 casEncryption attempted to set value to false Result: success
18:06:44 jfoInterface attempted to set value to eth1 Result: success
18:06:44 jfoEncryption attempted to set value to false Result: success
18:06:44 jfoWakeUp attempted to set value to 2000 Result: success
18:06:44 uavEncryption attempted to set value to false Result: success
18:06:44 casWakeUp attempted to set value to 2000 Result: success
18:06:56 uavWakeUp attempted to set value to 2000 Result: success
18:06:56 jtacWakeUp attempted to set value to 2000 Result: success
18:06:56 jtacEncryption attempted to set value to true Result: success
18:06:56 uavInterface attempted to set value to eth0 Result: success
18:06:56 limeCZConfig attempted to set value to 100 Result: success
18:06:56 uavCompress attempted to set value to HIGH Result: success
18:06:56 grapeCZConfig attempted to set value to 100 Result: success
18:06:56 grapeFWSPA attempted to set value to lenient Result: success
18:06:56 casEncryption attempted to set value to true Result: success
18:06:56 jfoEncryption attempted to set value to true Result: success
```

Figure 26: Actuation logs

Assessment Details: QIAAMU operates on a cause – effect network, which is a tree of nodes where each node represents parameters of the network. These parameters can be system conditions, actuators, or intermediate nodes which are a combination of system conditions, actuators, or other nodes. The value of the attribute and scope that the stakeholder cares about is derived from the nodes in the tree that are above it, also called ancestors. In order to find out more information about a particular assessment, that row may be selected from the assessment table in the first tab, and the 'Add Tab Showing Selected Assessment Ancestors' button selected. This will add another tab (see Figure 27), in addition to the 'Assessments' and 'Actuation Logs' tab, with the name of the attribute/scope. In that tab will be a table displaying a list of the assessment ancestors and the value associated with that ancestor. If the tab was added when the current mission mode was being viewed, the ancestor values will be the most recent values. If the tab was added when a previous mission mode was selected for viewing, the ancestor values will have the final value of that mission mode. Many of these tabs may be opened so that multiple attribute ancestors may be viewed, or the same attribute may be opened again at a different time so the differences over time may be observed.



Mission Mode: Image Annotation (showing values from the time this tab was created)

CZAvail Ancestors

Ancestor	Value
limeFWSPA	lenient
limeCZConfig	100
casCrumpleZone	grape
uavCrumpleZone	grape
jtacCrumpleZone	lime
jfoCrumpleZone	lime
grapeFWSPA	lenient
grapeCZConfig	100
limePingable	1
grapePingable	1

Figure 27: Assessment details: explaining why the requirements were deemed (un)met

4.5.1.2.2 Hex Viewer Integration

In the final year of the project, AFRL announced the availability of an advanced Air Space Cyber User Defined Operational Picture (ASC UDOP) visualization framework and associated tool set. One aspect of the framework, namely, the Hex Viewer (Priority Filter Viewer) is extremely synergistic with QIAAMU's runtime visualization.

QIAAMU presents the QoS and security assessment based on measurements and observations obtained from the system, and attempts to manipulate the system - the act of which changes these measurements and observations providing better satisfaction of stakeholder requirements. The assessment details panel of the QIAAMU browser-based runtime visualization presents near real time discrete snapshots of the assessment results and what system condition and actuator values contributed to that state. With Hex Viewer, it is possible to watch the cause-effect network as a whole with dependency information visually integrated into the display, i.e., see how changes in lower level observations and measurements affect the assessed value. The built in support for dependency propagation in Hex Viewer enables one to see slices through the complex network that are causally related.

We therefore developed an extension to QIAAMU runtime visualization to interoperate with Hex Viewer. The Hex Viewer represents measured and computed values as hexagons. The computed values can be organized in levels. How the higher level computed values depend on lower level values (and ultimately, at the base level measured values) is defined by node relationships. In essence measured values are propagated through the dependency network, updating the computed values, and based on predefined rules, the color of the updated nodes as well as tool-tip values are updated. Rules for color map, visible number of levels, frequency of updates etc. can be provided as configuration settings to the Hex Viewer tool. The dependency relationship between levels as represented by tables or views in the database can be defined by using a graphical editor.

In order to use the Hex Viewer tool for our purpose, we needed four things:

- Defining the schema and ASC-UDOP visualization configuration: ASC-UDOP needs to be told where to retrieve data from and how that data is related. We have defined a single underlying schema that can be used to support all missions and cause-effect networks. A second layer of schema is required to sit on top of the base schema and expose the data in the manner required by ASC-UDOP (namely one table per level of the cause-effect network, and one table per relationship between levels of that network). An instance of this second layer schema has been created that supports all cause-effect networks used in the various simulation demonstrations – which have a maximum of four levels. This could easily be extended to more levels, and simply requires the creation of new database views over the existing data, and additional configuration within ASC-UDOP. Given a number of levels, the process of configuration could be automated by creating the appropriate views and modifying the ASC-UDOP XML files, however this effort is currently manual.
- Initializing the underlying database schema with data representing the set of nodes and the node relationships: The initial data set depends on the actual nodes in the cause-effect network and how they relate to one another. The cause-effect network is unique for each mission which means some configuration needs to be undertaken to support ASC-UDOP's use for any given mission. The *blackboard parser* provides this capability. It is a separate program that must be run to initialize the database tables prior to running the mission. The blackboard parser program parses all of the blackboard .xml files associated with a mission, and pulls out all of the nodes of the cause effect network. In order to display properly in the hex viewer, the 'level' of each node in the network must be determined. The 'level' of a node depends on the depth of parent nodes a particular node has. A base node with no parents has a 'level' of 0, and a node which has one or more parents who are base nodes has a level of 1, and so on. Note that the base nodes that provide the measured values are the system conditions and actuators, and the attributes are the ends of the branches and have the highest level.
- Measured values: While the blackboard parsing extracts the base nodes, node levels, and node relationship information these are essentially shells that need to be filled with real values. Ideally, one only needs to input the base level values, the rest of the values are obtained by propagating the base values according to predefined rules. In QIAAMU, the values of all nodes (measured and computed) are actually known by the union of all blackboards. To avoid duplication of computation that has already been paid for, we simply extract the values for all nodes in the cause and effect network and put that in an SQL database (MySQL). This is done by an extension of our Visualization Server, which has access to the log messages that indicate when values in the cause and effect network have changed. This makes our node relationship tables much simpler; it only needs to capture who depends on what, not how or how to propagate changes.

- Color map, source setting of the measured values etc: The color map for node values and color changing thresholds, the source database connection information, auto-update frequency, and how many intermediate levels of relationships are made visible are all done by setting specific configuration parameters of the Hex Viewer.

Figure 28 shows how the Hex Viewer tool is used for runtime visualization of the global cause and effect network, along with its relationship with the QIAAMU Visualization Server and the blackboard parser. In order to run the blackboard parser, the MySQL database service must be started first. This is done by executing the command '`service mysql start`'. If the database has not been created yet, a database named `qiaamu` should be created with a user "proxyuser" and a password "proxy".

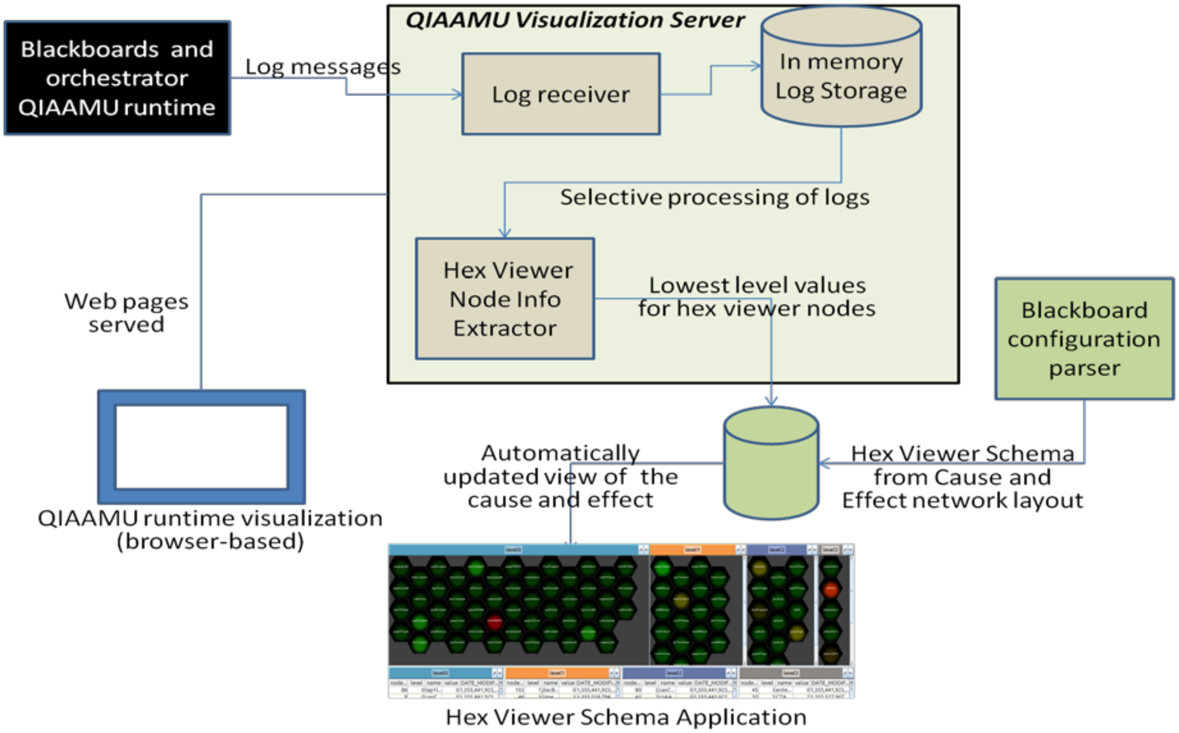

Blackboards and orchestrator QIAAMU runtime
Log messages
QIAAMU Visualization Server
Log receiver
In memory Log Storage
Selective processing of logs
Hex Viewer Node Info Extractor
Lowest level values for hex viewer nodes
Blackboard configuration parser
Web pages served
QIAAMU runtime visualization (browser-based)
Automatically updated view of the cause and effect
Hex Viewer Schema from Cause and Effect network layout
Hex Viewer Schema Application

Figure 28: Hex Viewer Integration

The node values provided by the Visualization Server are stored in database tables. Before storing values, the tables themselves and views into the tables then need to be initialized. This can be done by executing the SQL code in the NodeDatabaseCreation.txt file. Then the blackboard parser can be run. The parser is located in the project qiaamu-parse-blackboards, with a main class of parseAllBlackboards. An argument is required which is the full path to the directory containing the blackboard specifications to be parsed. As mentioned before, the information in the database is structured into two tables, the Node table and the Node Relationship table. The Node table contains the name, id, level, and value of a node. There will be one entry in this table for each node in the cause effect network. The Node Relationship table contains the information which links nodes together, the node id of a node and the node id of one of the parents of that node. A node will have one entry in this table for each of its parents. If a node has no parents, it will not have any entries in this table. The output of blackboard parser partially fills in the node relationship table and partially (all but the node value) fills in the node table. The value field is provided at runtime by the Visualization Server.

4.5.2 Assessment and Tradeoff Management within a Survivability Architecture

In order to perform runtime assessment and tradeoff management, the blackboards, the orchestrator and the visualization server must run within the distributed system being managed. Technically, the orchestrator and the visualization server can run on a separate host that is not participating in the mission essential functions, but they need to be accessible from the hosts running blackboards. For simplicity, we assume that these two components are placed in server class machines in the distributed system participating in the mission.

Clearly, the QIAAMU components and interaction among them needs to be protected as strongly as the business components and their interactions of the underlying distributed system. In other words, the QIAAMU components need to be integrated within the security and survivability architecture of the underlying system. To illustrate the issue, let us consider that the business and management interaction between host A and host B in the system is protected by Single Packet Authentication (SPA) and host to host TLS. If host A and host B runs blackboards, their interaction must also be subjected to the same level of protection, otherwise the introduction of QIAAMU components will undermine the overall security of the system. Another reason for integrating with the security and survivability architecture of the existing system is to take advantage of the sensors and actuators that become readily available to QIAAMU. Within a host, QIAAMU processes are run under a distinct user identity that can be used to control the privileges granted to them on a host by host basis.

QIAAMU blackboards interface with other elements of the distributed system, OS and resource management mechanisms, security mechanisms as well as application components. The blackboard's interaction with other elements of the system are of two kinds: some are read only- these are used for obtaining measurements and observations from the system into QIAAMU, and some are read-write—these are used to set values into configurable actuation control mechanisms in the system. This implies that blackboards that are primaries for actuators can have quite high level of privilege (including for example, sudo or JMX access to manipulate JVMs). This is inevitable if we would want to drive autonomic adaptation based on runtime tradeoff analyses. However, recall that our guiding principle for blackboard placement is that an actuation control and the blackboard that has the primary responsibility to interface with that actuation control are to be collocated. This makes it possible to reign in the risk of unmitigated privilege dispersion—a compromised blackboard process can only manipulate the actuators that are collocated on that host.

For our internal testing and evaluation and demonstration, we have inserted QIAAMU components into the APS survivability architecture, which is used to protect a services enclave. The services enclave offers a critical information management service that the mission participants use to interact among each other. More about this integration appears in Section 4.6.2. Installation procedure and how to use the prototype for runtime assessment and tradeoff is described in a separate User's Guide document (CDRL A009).

4.6 Experimental Evaluation

A mission scenario executing in a distributed environment is essential for evaluating QIAAMU. In the initial part of the project we used a COREmu based emulated environment and MRAP (MRAP stands for Mine Resistant Armored ..) route clearing mission. In this mission MRAP vehicles go out to ensure a route is cleared of explosives. If they encounter suspicious activity and improvised explosive devices, they need to immediately provide visual and other information to the Headquarter (HQ). There are different ways to communicate with the information system at the HQ, the MRAPs can use line of site communication with their forward operating base (FOB), or they can use satellite communication. Line of sight and satellite communication offers different levels of QoS and security. For the internal test and evaluation and final demonstration, we used a virtual machine based environment running a close air support mission. In this mission, images taken by a loitering UAV is disseminated to tactical users via a publish-subscribe-based information management system. The tactical users annotate selected images and republish them, and for selected targets, issue a call for fire. The call for fire and annotated images are received by the attack aircraft that publishes read back and confirmation of targeting information. The tactical users have the option to call off or confirm the target until a certain time, beyond which the mission is a go and cannot be called back.

4.6.1 Early Evaluation

Immediately after the initial prototype of the blackboards was developed, we undertook a performance evaluation to understand the following:

- The gross impact of QIAAMU infrastructure on the application: the overhead added to the application by the QIAAMU components, measured by increased round trip latency.
- The cost of updating system conditions, i.e., the cost of getting measurement and observation data from ESM databases to the components that use the data in assessment and response decisions
- The cost of performing *assessment* i.e., the time spent in determining whether the system is delivering the desired level of IA and QoS based on stakeholder-specified requirements and the measurements and observations obtained from the system. This is measured by the elapsed time between two timestamps T1 and T2, where T1 marks when the blackboard initiates an assessment and T2 marks when the assessment completes.
- Cost of tradeoff based response: the cost of adjusting the system's configuration, guided by stakeholder specified preferences, when not all requirements can be met.

A technical report documenting the results of this preliminary evaluation was delivered as an interim evaluation report in 2011. In this section we briefly describe the test environment and summarize the key results.

4.6.1.1 System under Test Used in the Early Evaluation

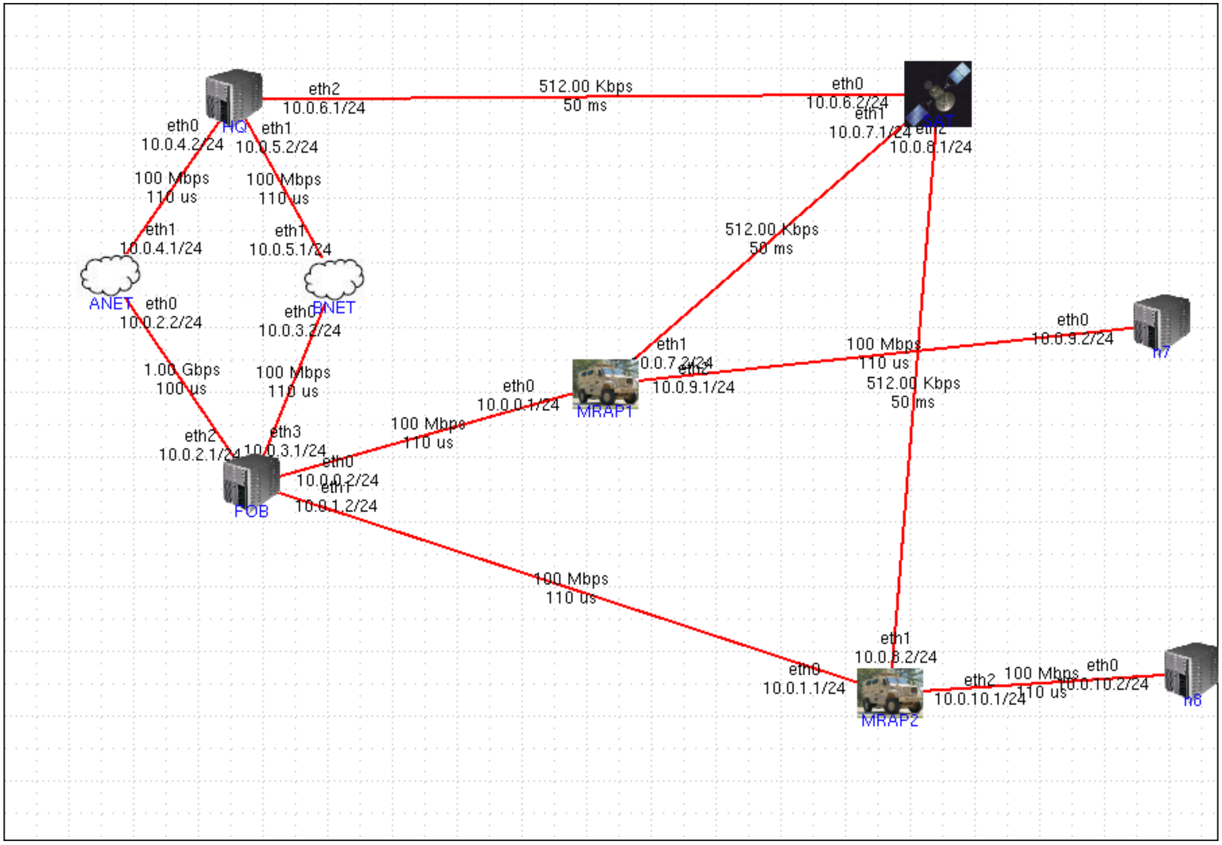


Figure 29: System under test (emulated network with QIAAMU components)

To estimate the performance cost of the key QIAAMU operations described above, we performed repeated experiments where those operations take place and collected performance data. These experiments were performed in an emulated network, which was subjected to various injected events to trigger assessment and reconfiguration. The entire emulated network, implemented using COREEMU [43], run on a single, four processor (Quad core Xeon 1.60 GHz) Linux server with a 2G RAM. We note that the hardware that we currently have available is at the low end of the recommended CPU/memory resources by the COREEMU developers, which imposes certain restrictions on what can be done in the emulated environment as well as impacts the performance of the QIAAMU components.

The emulated network (see Figure 29) consisted of nine CORE containers. Each *container* is a virtual machine. A container can represent a network cloud (as ANET and BNET in the figure) or a node in the network. A node in the network can be a single host, or an abstraction of a small office or campus connected to the emulated network.

We used two illustrative Enterprise System Management (ESM) mechanisms, OSSEC and Nagios to demonstrate integration of ESMs as the sources of measurement that QIAAMU assessment can use. The OSSEC *server* on the Linux server host, outside the emulated network, but the Nagios *server* ran on an emulated node (specifically, the FOB) —this illustrates that QIAAMU can integrate with existing ESM servers that ran on a local host (i.e., the QIAAMU component and the ESM server process run on the same node) as well as ESM servers that are remote (i.e., the ESM process runs on a host that does not run any QIAAMU component).

Every container ran a small set of processes, which were started after container initialization. These processes included SSH and SNMP servers, an IPERF server, and an OSSEC agent. SSH is used for administrative login into the container. SNMP is used to report container status to Nagios. IPERF is used as a representative of probing and measurement that is not part of ESMs – we fully anticipate extra-ESM measurement and observations will be needed for QIAAMU assessment in a real life situation.

The application scenario used in the experiments involved the MRAP1 trying to communicate with the HQ, simulated by a client in MRAP1 and a server in HQ where the client sends a short message (less than a KB) to the server every 10 seconds. The server at the HQ sends a message back in response; and the client logs the round trip response time.

The experiments also made use of a traffic generator, consisting of a client and server pair that can be selectively deployed in two CORE containers to simulate flooding or background traffic in the link between the two containers. In our test three simultaneous threads with a delay of 1 millisecond between sends in each thread and a packet size of 2048 was able to saturate the link between MRAP1 and FOB.

QIAAMU blackboards were launched on all the VMs except two (n6 and n7 in figure 29). These blackboards were loaded with the following baseline configuration, which was subsequently modified for various specific experiments:

- HQ - 6 Assessments, 34 System Conditions, no peering
- FOB - No assessments, 21 System Conditions, no peering
- MRAP1- 2 Assessments, 15 System Conditions, no peering
- MRAP2 - 2 Assessments, 16 System Conditions, no peering
- ANET - No assessments, No system Conditions
- BNET - No assessments, No system Conditions

4.6.1.2 Key Findings of the Early Evaluation

Impact of QIAAMU on application: We ran the route clearing mission scenario with and without the six QIAAMU blackboards described, and measured latency of application messages. In this run, the MRAP1 client was sending short messages every 10s. The blackboards were set to refresh there system conditions from the ESMs at every 60s. The runs with the blackboards also had varying degrees of peering. The results show that as peering increases the application latency also increases. It confirms the intuitive hypothesis that peering increases the amount of inter-blackboard messages, which load the network, which in turn increases the application latency. As shown in Figure 30, without any peering the application latency with and without blackboards is really indistinguishable. With peering, the application latency increases, but the absolute amount of the increase is really small (from 6.34ms to 6.41ms).

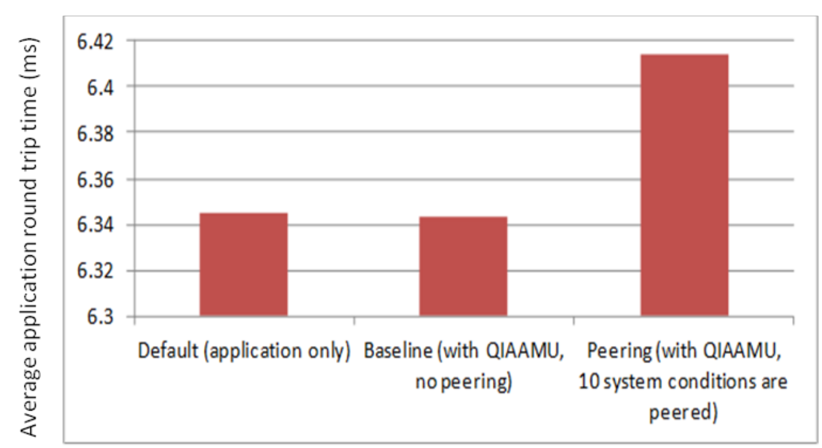


Figure 30: Impact of QIAAMU blackboards and inter-blackboard interaction on application latency

Cost of system condition update and peering: Experiments showed that with or without peering, the time to refresh the system conditions in a blackboard has an initial spike (see Figure 31). This has to do with the initialization of the blackboard and establishing connections to ESM databases and other sources of raw measurements. With peering, the time to refresh the system conditions increases—this is also to be expected. Recall that the source of a raw measurement and the primary blackboard for that measurement are collocated. Peering that system condition incurs the cost of a more expensive remote call with another backboard instead of a local read.

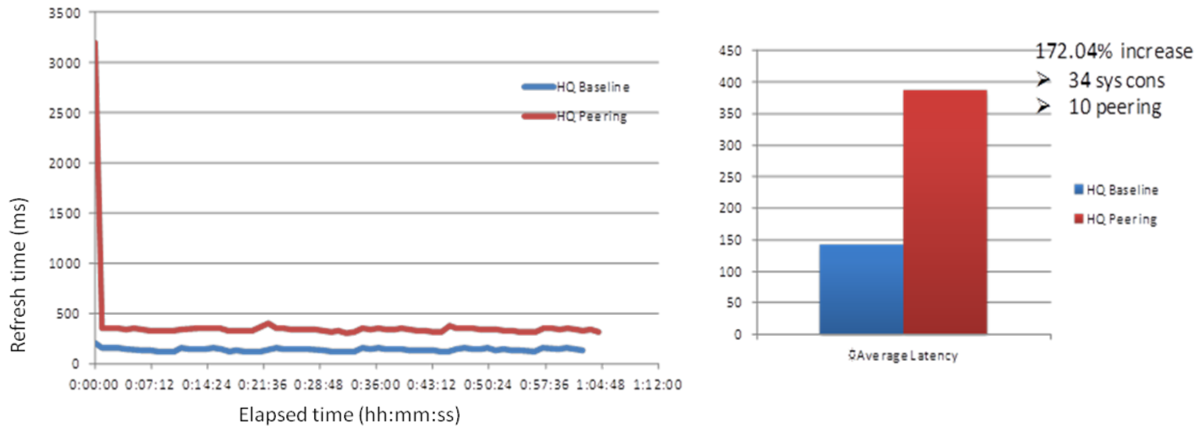


Figure 31: Estimating the time to perform system condition refresh in a blackboard

Time to perform assessments: Figure 32 shows a plot of such raw assessment times in one blackboard with 6 assessments, 34 system conditions. In these experiments a fixed pattern of application and background traffic was used in all runs, and runs alternated between 0 and 10 system conditions being peered. The number of assessments performed by the blackboards varied during runs. The assessment time is in milliseconds, and although the assessment time stayed within 10ms, there are visible spikes associated with initial set up.

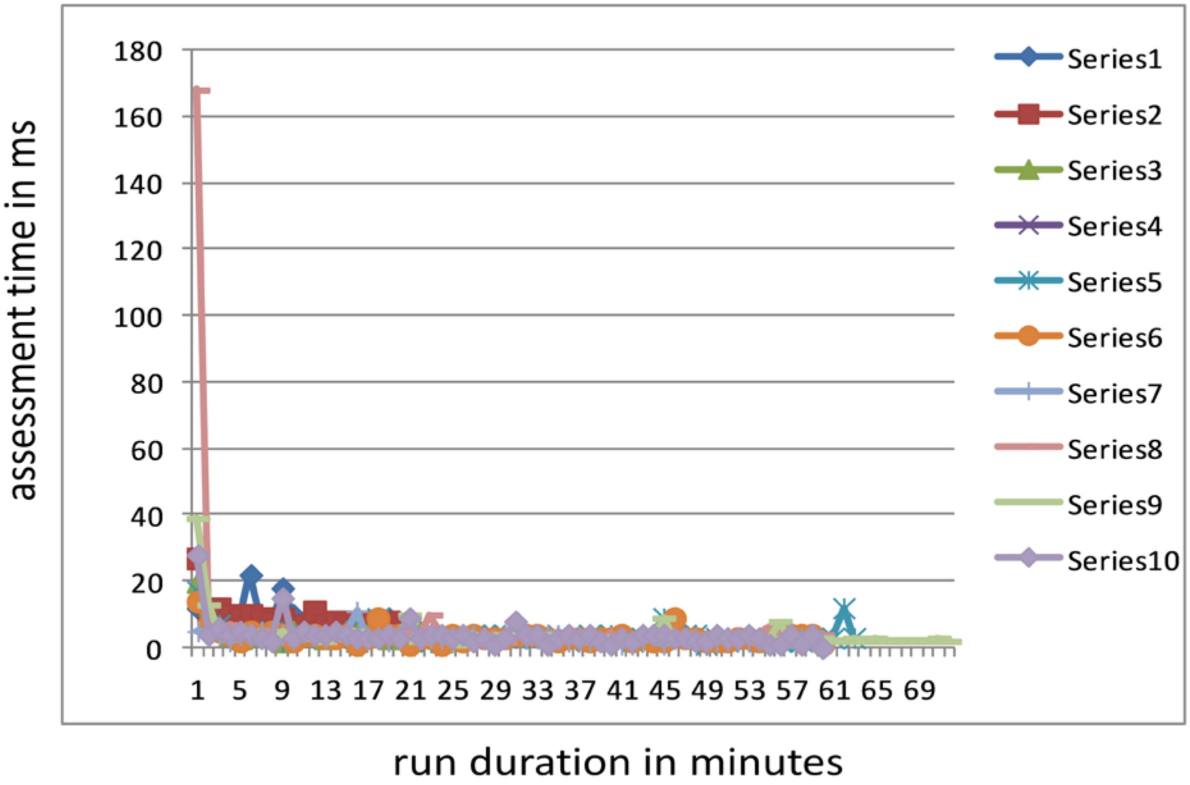


Figure 32: Estimating the time to perform assessment computation

Tradeoff analysis and response time: At the time of the initial evaluation, we only had a simple proof of concept actuator (selecting which network interface the MRAPs should use), and a simple tradeoff management algorithm based on predefined preferences. For instance, if the mission mode prefers performance over security over bandwidth, and the system is using a higher security low bandwidth network, QIAAMU tradeoff will change the network interface to use the other interface that connects to the higher bandwidth lower security network. In this experiment, flooding was used to reduce the effective bandwidth on the high bandwidth lower security network. QIAAMU assessment detected that the bandwidth requirement is not met, and therefore switched to the other network, which offered higher security but lower bandwidth. The next cycle of assessment detected that this lower bandwidth network is providing better bandwidth than what the flooded higher bandwidth network and settles on the interface choice. Clearly, the number of steps required to settle down depends on the initial starting configuration. In the worst case, the prototype required two assessment intervals, with intervals set to ~60 seconds, to come to a steady state. The average actuation latency will vary with the type of actuations being performed. That being said, the interface change actuation used in the evaluation scenario is never more than 200 milliseconds.

4.6.2 Internal Test and Evaluation (IT&E)

In the IT&E experiments, we use various flavors of a close air support mission scenario that was developed by another AFRL project, namely, video to JTAC. The video to JTAC scenario is an example of ground forces directing an aircraft providing Close Air Support (CAS) via video and image annotation. Multiple actors are publishing video and image data of a potential target. A JTAC on the ground annotates this media indicating a target or other important features and then publishes the annotated images. The CAS aircraft indicates that it received the annotated image at which point the JTAC publishes a full CAS Briefing describing the desired strike parameters. The CAS Briefing is "read back" to the JTAC to confirm it is correct and as the CAS aircraft approaches the target it publishes its targeting data. The JTAC then forwards a hot or abort message to the aircraft. Additionally in this scenario there is a Joint Fire Observer (JFO) who has subscribed to both the CAS briefing published by the JTAC and the targeting data from the CAS Aircraft. The JFO is capable of sending an abort message to cancel the strike, for example if they are aware of friendly forces in the strike area.

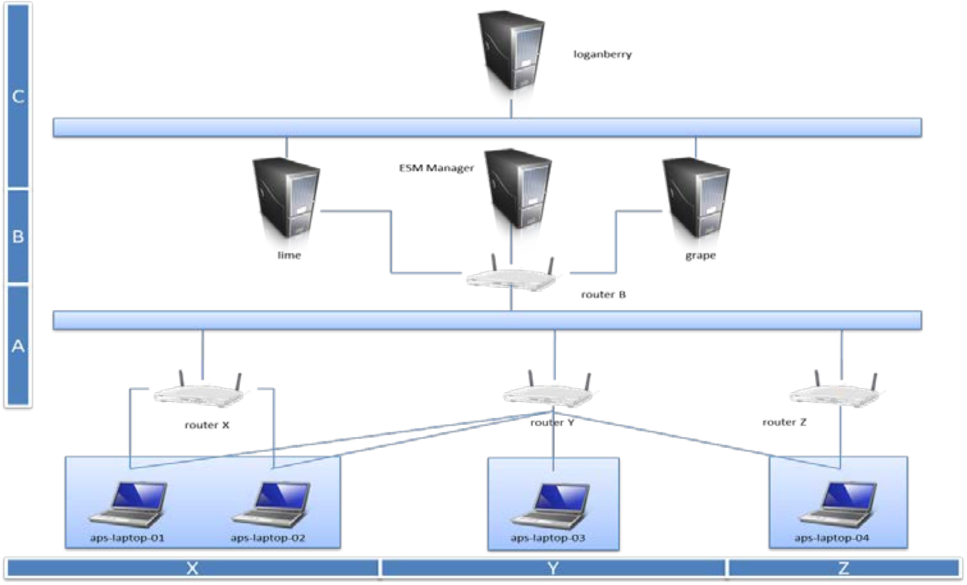


Figure 33: System under Test

As can be imagined, the various phases of this scenario come with varying effects on the underlying network and systems (i.e. during image acquisition the network and servers may have high levels of traffic as they route and process the incoming media). These natural variations in the state of the system in conjunction with the effects attributable to an adversary present in the simulation (who undertakes various actions depending upon the details of a particular scenario) serve to exercise the various actuation instruments under the guidance of QIAAMU runtime assessment and tradeoff management.

We implemented software components that emulate the behavior and actions of the various mission stakeholders, some with interactive GUIs that can be used for demonstration and controlling the mission progress. These stakeholder applications interact with each other using a publish-subscribe based information management service (IMS) implemented in JBoss.

The distributed system consisting of the IMS and the stakeholder applications are protected using the survivability architecture developed in the Advanced Protected Services (APS) project. The IMS and the stakeholder components run different network enclaves. The network enclaves are virtualized; the hosts running the IMS, the stakeholder applications as well as other hosts needed by the APS architecture are all virtual machines. Multiple different variations (e.g., with different starting configuration or slightly changed stakeholder requirements) of the basic video to JTAC mission were run in this distributed system context.

The system under test, experiment methodology and results are described in the Internal Test and Evaluation document. In the rest of this section we will provide a brief and high level summary.

4.6.2.1 System under Test

Figure 33 shows the system under test. Networks B and C represent the enterprise networks hosting critical services. Network A is the wide area access network. Networks X, Y and Z represent tactical networks. The JTAC application runs on laptop-01, UAV on laptop-02, and CAS on laptop-04. The hosts lime and grape run a redundant crumple zone, protecting the high valued IMS running in the host loganberry. All networks and hosts are virtualized, and the entire setup was run on a BBN cluster node (Dell R710 with dual 6 core AMD Opteron processors and 32 GB RAM). Figure 34 shows the test set up highlighted with the defense mechanisms organized in a survivability-focused architecture. Note that although JBoss (at the core of the IMS) can be used with both http and https, in the experiments, we used only http and application level encryption options.

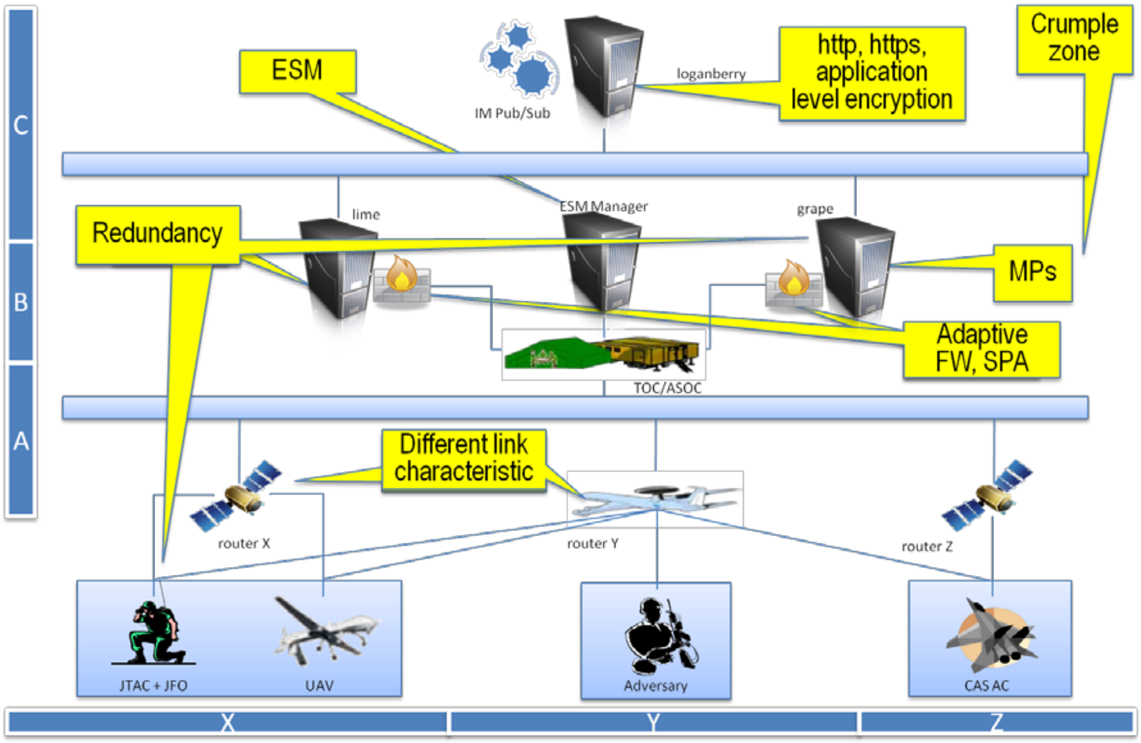


Figure 34: Test setup with annotated defenses

The system under test provides the following measurements and observations (represented as system conditions to the blackboards):

- Via ESM (i.e., blackboards obtain these measurements by accessing ESM databases)
  - Remote Server CPU (as provided by Nagios)
  - JBoss Application Server Memory (as provided by Nagios)
  - APS Attack State (from the APS log analysis database)
  - Remote Reachability (as provided by Nagios – used when the source node is not a blackboard)
  - Oval Scan (as provided through OpenSCAP via OSSEC)
- Via custom instrumentation (i.e., blackboards obtain these measurements and observations from special purpose instrumentation/agents inserted into the system)
  - Reachability (source node is a blackboard)
  - Point to Point Bandwidth
  - Host CPU Usage
  - Crumple Zone Configuration
  - Firewall Configuration
  - Announced maintenance (i.e., whether a host is under pre-announced maintenance)
- Via JMX (i.e., blackboards make use of JMX to obtain information from JVMs)
  - Client Compression Level

    - Client Encryption State
    - JNDI Profile Selection
    - Network Interface Selection
- Others (Readily Available from the OS) (i.e., blackboards invoke OS features to obtain these measurements and observations)
    - Process Alive
    - Process Memory
    - Syslog

The system under test provides following actuators – i.e., mechanism and features that can be changes at runtime:

- Via custom Instrumentation
    - Crumple Zone Configuration
    - Firewall Configuration
- Via JMX
    - Client Compression Level
    - Client Encryption State
    - JNDI Profile Selection (which crumple zone to use)
    - Network Interface Selection
    - Crumple Zone selection

In addition, the test system is equipped with a number of failure injection mechanisms and an access point for mounting attacks. The attacker access point (laptop 3) is connected to the access network (network A), but does not have any privilege to access the IMS.

4.6.2.2 Experiment Methodology

The test methodology we adopted was primarily based on fault injection. Instead of actually developing attacks and mechanisms to subvert the security mechanisms in the system, we simply provided the required access and privilege to injection mechanisms that can perform what the adversary intends. Such injection mechanisms can be invoked in a scripted manner, which is useful for performing repeatable runs and collecting metrics, as well as manually, which is useful for demonstrations.

The evaluation included repeating the same experimental runs under two different configurations. In one configuration, QIAAMU was allowed to perform the tradeoff actuations when necessary, and in the other, assessment, tradeoff and penalty computation was performed, but QIAAMU was not allowed to actuate. This enabled us to get a better understanding of what would have happened to the mission without QIAAMU. We also performed experiments to estimate the resource overhead of QIAAMU. These experiments focused on the overhead to each of the VMs as well as the total load on the physical host with and without QIAAMU.

The injection scripts focused on emulating a range of attack effects and failures including DoS focused resource exhaustion attacks (e.g., CPU and bandwidth exhaustion), crash failures including link and node failures and process deaths, and privilege escalation corruption attacks such as creating backdoor accounts in critical assets.

A few comments about the injection mechanisms and QIAAMU measurements and observations are in order. The list of provided injection effects are only a sample, and by no means the exhaustive list of effects that can be injected into the system. While it is easy to construct and add new injection effects, whether the existing ESM and custom monitoring mechanisms will pick up and detect such injected effects is an entirely different and independent question that depends on the security architecture of the underlying system. QIAAMU does not design the defense—and is not responsible for choosing the security mechanisms in the system. However, given the defenses and system management mechanisms present in a system, there is quite a bit of latitude to devise the system conditions projected to the QIAAMU blackboards. These can be achieved by processing and aggregating the raw measurements and observations provided by the existing defenses and system management mechanisms. QIAAMU provides an XML based framework to aggregate and extract derivatives from the raw measurements and observations available in the system. In addition, if permitted, additional custom instrumentation can be added to watch for and collect additional pieces of information.

Using the provided set of injection effects, we studied a disaster situation (that utilized system redundancy to survive), and a maintenance scenario (where unannounced maintenance gets in the way of the mission).

4.6.2.3 Experiment Metrics

The following experimental metrics were computed from the data collected from the experimental runs. The objective was to gain a better understanding of the positive contribution runtime assessment and tradeoff makes as well as the overhead cost incurred.

4.6.2.3.1 Effectiveness

The purpose of actuations is to reduce the penalty incurred by the system, and the effectiveness metric measures this by calculating the *average penalty reduction*. Recall that penalty is the aggregate deviation from the desired level of QoS and Security for all stakeholders, and is readily available from the QIAAMU infrastructure.

$$E = \frac{\sum_{i=1}^{n} \max(P_{\text{before}}(s_i) - P_{\text{after}}(s_i, a_i), 0)}{n}$$

There are n tradeoffs, P() is the penalty, s_i represents the system conditions at time i, and a_i represents the result of the actuations at time i. If the penalty after a tradeoff is less than the penalty before the tradeoff, the system will actuate. Otherwise, the configuration settings proposed by the tradeoff will not be committed (hence the max function). Because there are no actuations in the “noact” runs, this metric has no meaning for those runs.

4.6.2.3.2 Response Time

We measure the time from when the attack occurs to when QIAAMU has finished implementing the new actuator configurations to mitigate the effects of the attack. QIAAMU's response to attacks is not always clear (sometimes it decides not to actuate at all, for example), so we have only reported response times for scenarios in which QIAAMU makes a clear move in response to an attack. The response time is also measured from the start of the attack injection activity and ends when a penalty reduction is confirmed (see Figure 35). This is also measured as the outermost envelope for estimating the time taken by QIAAMU runtime machinations because sometimes multiple engagement is required for attack injection (e.g., ssh into a host, copy or execute a script etc), and before the penalty reduction is confirmed, tradeoff computation must be triggered by one or more assessments indicating some requirements are not being met.

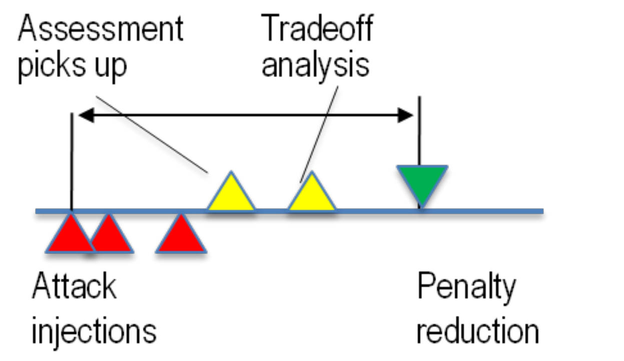


Figure 35: Explanation of response time

4.6.2.3.3 Time-Average Penalty

The purpose of using QIAAMU is to minimize the penalty incurred by the system. The average penalty is calculated by integrating the penalty over the entire mission time, and dividing by the total mission time. This measure is essentially a normalized version of the area under the penalty curve—the smaller it is the better is the value provided by the QIAAMU runtime assessment and tradeoff management.

4.6.2.3.4 Application Latency

At the application level, pub-sub messages are being sent between the participants. We measure the average latency of these messages (the subset that reach CAS). This metric gives us insight into the effect of QIAAMU's decisions at the application level, as opposed to the first three metrics, which measure QIAAMU's performance itself.

4.6.2.3.5 Penalty Plots

Time average penalty can present a misleading picture if the mission ran for a long time and spent longer time in the low penalty region. For this reason, we also show the plot of penalty over time as the mission progresses. In these plots, the x axis is time in seconds, and the y axis is the penalty (which is just an integer). In the horizontal axis we indicate mission mode changes by a gray vertical line. Injection of failures and attacker activity is shown as a red vertical line. The penalty line is normally painted in blue, but becomes green for the region where the result of tradeoff driven actuation reduced the penalty. The color changes to blue again under the following two conditions: (1) the penalty changes or (2) tradeoff driven actuation is not engaged by QIAAMU because the suggested actuations would have increased the penalty. Change in penalty (upward due to attack or normal resource usage during the mission) is shown in black.

4.6.2.3.6 Overhead Metrics

In addition, we performed experiments to estimate the added overhead incurred by QIAAMU. The goal of these experiments was to record the CPU and memory usage of the VMs and the physical host during the scenario runs with and without QIAAMU.

4.6.2.4 Evaluating the Efficacy of Runtime Assessment and Tradeoff Management

We have two ways to run the scenario and subject a run to effect injection. The first one is fully scripted, where the mission progresses automatically without any operator invention, i.e., stakeholder applications follow a scripted set of activities which cause the mission to progress forward. In addition, effect injection is also scripted, i.e., at a programmed time point within the mission, the intended effect is injected into the system. The effect injection typically involves multiple steps, such as logging into a host, uploading a script or executing a pre-loaded script etc. This mode of operation is good for repeated runs and collecting data. The other mode of operation is an interactive one, where the stakeholder applications are controlled by interactive GUIs and mission progress depends on the actions performed by human operators through these GUIs. Effect injection is also done manually, i.e., when desired, the operator can perform login and execute the scripts necessary to inject the effects. This mode of operation is good for demonstrations because the pace and effect injection can be controlled by the demonstrator.

4.6.2.4.1 Local (Dry) Runs

Before running the scenarios and subjecting the run to injected effects in the VM-based system under test, we performed dry runs of the scripted run to assess the outcome of VM runs. These dry runs involved running all blackboards in a single host and providing realistic values for the system conditions that were recorded from the system under test previously. These dry runs are much easier to run than the running the system under test, and can be run in office desktops instead of a large cluster node. We run much more of these than the actual VM runs to eliminate blackboard configuration bugs and tuning the effects injections. We selected a handful of these runs to run in the VM based system under test to collect experimental results.

4.6.2.4.2 VM Runs

The IT&E report documents the results of various experimental runs in detail. In this section we will simply use a pair of runs as an illustrative example: in the first run QIAAMU assessment and tradeoff management was allowed to make changes in the system, and in the second, the so called "no actuation" run, QIAAMU is used to compute the penalty value only. The idea is that by comparing the former with the later "no actuation" run, we can envision what would have happened to the mission without runtime tradeoff-driven adaptation.

In the first of the pair of runs, all clients initially use a default value for which crumple zone (CZ) node to use. QIAAMU blackboards utilize the opportunity to improve QoS by load balancing the concurrent clients among the two available CZ nodes. During mission mode 2, an attacker gains access to lime (one of the CZ hosts) and kills a service on that node. This service is not directly a part of the pub-sub operation—so pub-sub operations can still go through lime, instead the killed process is representative of auxiliary services that are critical for ongoing system and security management. In this experiment we used `crond`, a kernel service as an example of such a critical service. Obtaining process death information for registered/designated services from the OS is relatively straightforward. The anomalous process death is a cause for suspicion, especially if the host concerned is not scheduled for maintenance. In this run, no maintenance was scheduled overlapping the mission, so lime was assessed to be untrustworthy: the dead service is a strong indication of malicious attack and the system incurs a very high penalty because some clients are using an untrustworthy CZ node. Tradeoff driven actuation switches all clients to use the other CZ node in order to reduce the penalty—as shown in Figure 36. This tradeoff is an example of trading off QoS for security—all clients are now served by the single CZ node—a QoS condition that the earlier tradeoff action tried to make better, however this lower QoS is preferred because in the present mission mode security is more important than higher QoS.

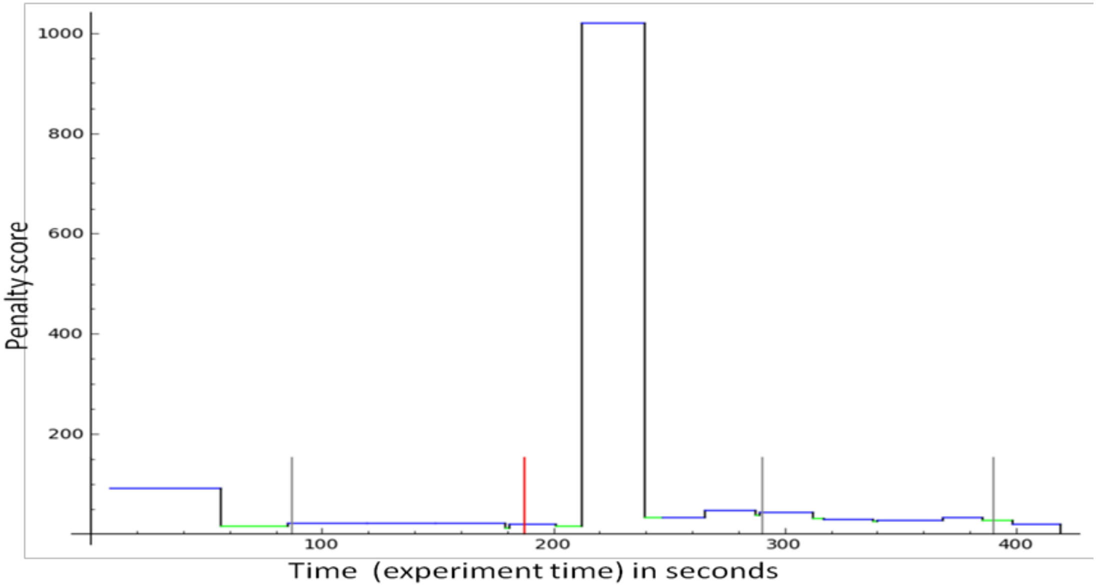


Figure 36: Penalty plot for the run with anomalous process deaths in the crumple zone

The effectiveness metric for this run was 86.5, time average penalty is 120.07, the reaction time (for responding to the process death attack) was 52 seconds and the average application latency was 3018 ms. Comparing the metrics and the penalty plots with the calibration run, this run shows a higher effectiveness, primarily because this run has more opportunity to reduce penalty by doing tradeoff (note that the calibration run had no attack). Apart from the attack induced spike, the penalty plot roughly follows the same pattern. The application latency saw a moderate increase from ~2000ms to ~3000ms. Finally, this run took 52 seconds to recognize the attack effect, perform the tradeoff computation, set the tunable configurations and confirm the penalty reduction. To answer the question "what would the situation have been without QIAAMU?" we need look at the next run, which consisted of the same attack launched at the same time within the mission—but in this run QIAAMU actuation was disabled.

As mentioned earlier, this kind of "no actuation" runs allow us to demonstrate how effective QIAAMU is and what the situation would be like if the distributed system did not take remedial actions during attacks.

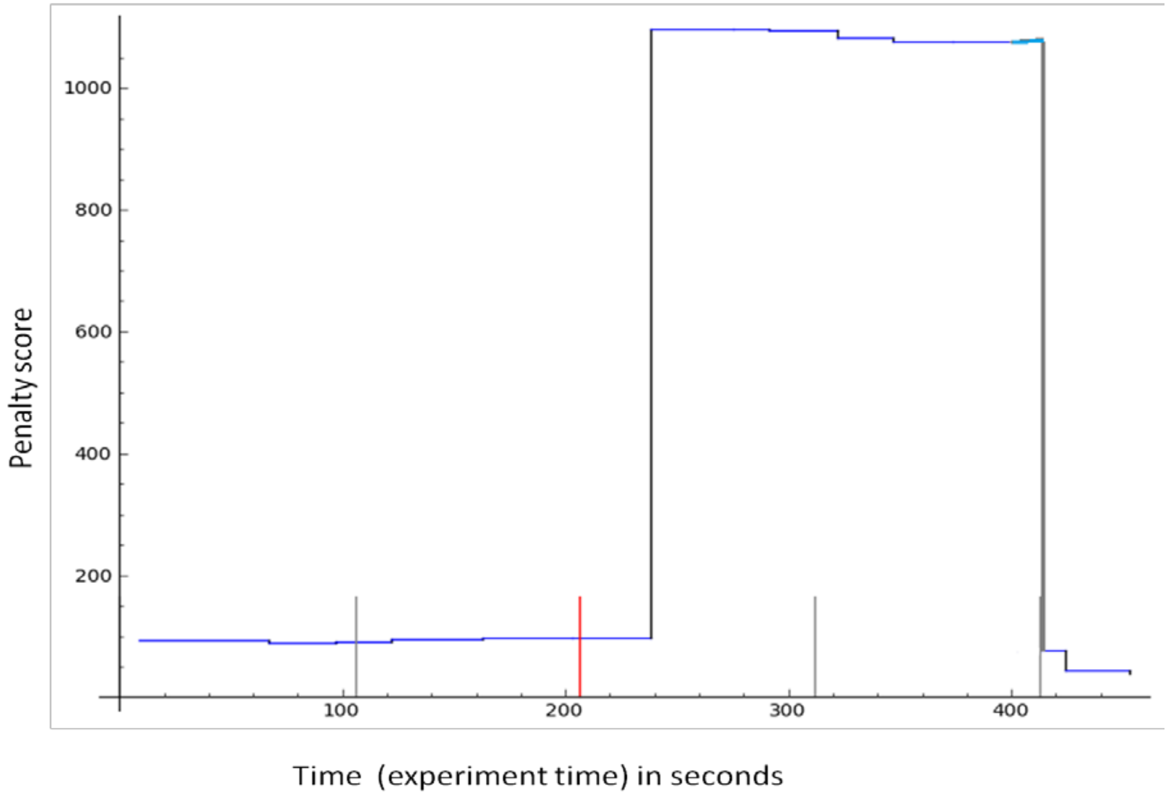


Figure 37: Penalty plot for anomalous process deaths in the crumple zone with "no actuation"

As shown in Figure 37, the penalty shot up at the same place relative to mission timeframe as in the previous run after the attack injection, but remained high for a longer time. Also, since there was no actuation all clients stayed with the CZ node they started with —thus we did not see the initial penalty reduction due to QoS improvement resulting from the load balancing response. The penalty changed only when the requirements changed (at mission mode change).

The time average penalty is much higher (~567) compared to the run with actuation (~120), and the unmitigated QoS impact is also visible in the application latency (~33000ms, compared to ~3000ms). It should also be noted that if a host in the managed enclave is found to exhibit an unexplained anomaly, under the current doctrine the knee-jerk reaction would have been to shutdown the enclave—which would make the IMS unavailable, which in turn would cause the mission to stop. With the unified view of the mission's QoS and security requirements and the security and QoS state of the distributed system enabled by the QIAAMU assessment and tradeoff, mission essential functions can still be allowed to continue with a level of confidence that is currently unachievable.

4.6.2.5 Evaluating the Overhead

Analysis of the data collected from the experiments performed with the objective of assessing the overhead of QIAAMU runtime operations indicate that QIAAMU has a low impact on its environment, both in terms of CPU and memory usage.

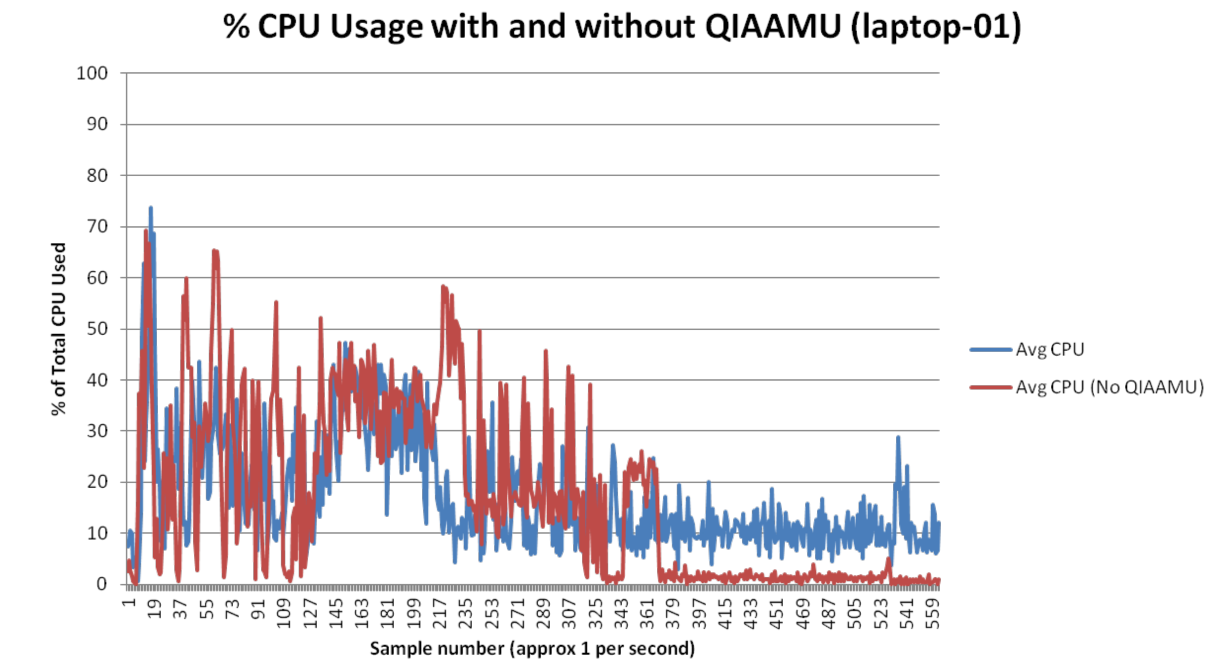


Figure 38: % CPU usage with and without QIAAMU for laptop-01

As can be seen in Figure 38 and Figure 39, QIAAMU may add a roughly 5-30 percent CPU overhead to an otherwise idle node, but during periods of machine activity, the CPU signatures between QIAAMU and non-QIAAMU runs on average are practically indistinguishable (approximately 1/3 of one percent difference in the tests executed). Differences in memory consumption with and without QIAAMU may appear more prominent in the charts. However, given the small amount of total memory on each virtual machine, these differences only amount to approximately 103 MB per VM on average.

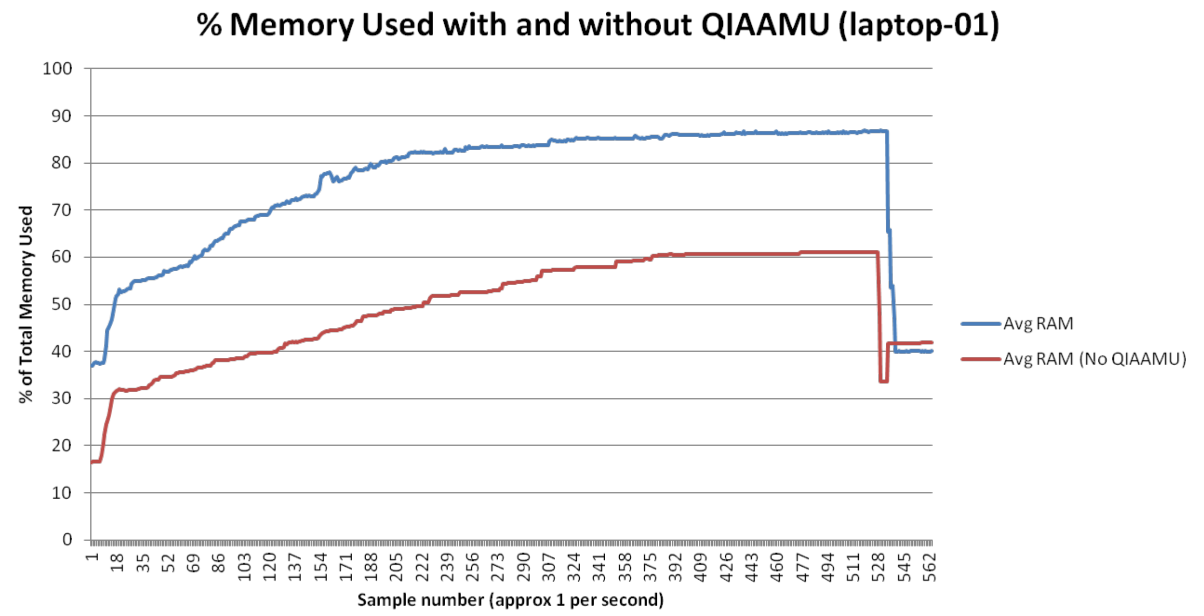


Figure 39: % Memory used with and without QIAAMU for laptop-01

Similarly, looking at the physical cluster node hosting the virtualized system under test, we see that QIAAMU components running on various VMs incur a reasonable overhead. With QIAAMU, the physical host's CPU consumption (see Figure 40) was averaged near 30% throughout the mission. Without QIAAMU, the CPU load introduced by running scenario varied from very low to as high as 30%. This is to be expected, because depending on the mission mode, stakeholder applications were computing and communicating more or less, whereas, QIAAMU components needed to compute and communicate pretty regularly irrespective of what mode the

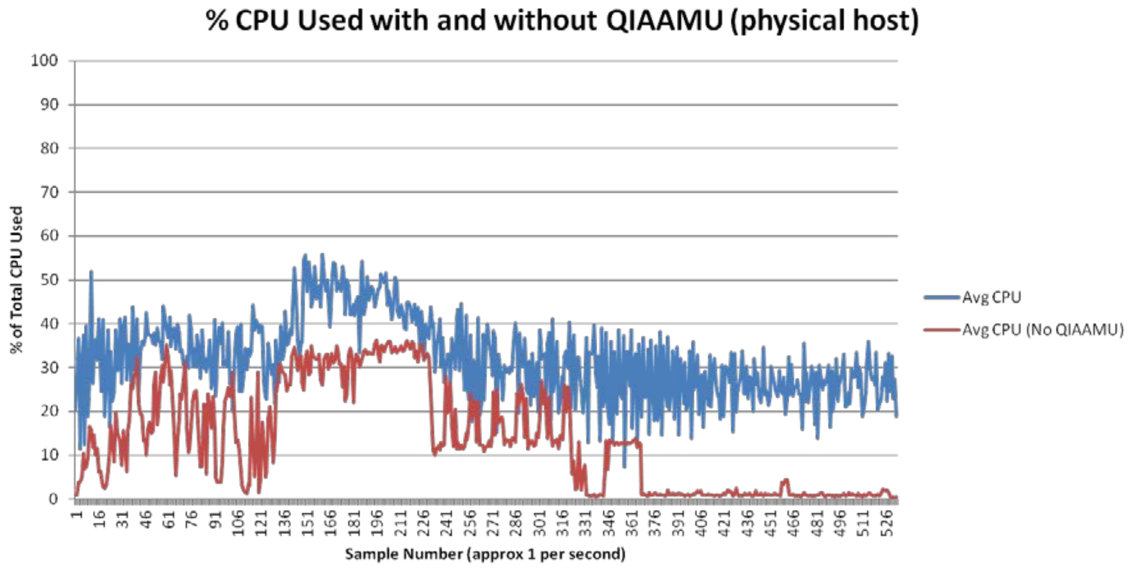


Figure 40: % CPU used with and without QIAAMU for the physical host

mission is in. The memory consumption chart in Figure 41 shows that with QIAAMU, the physical hosts memory consumption grew upward and the flattened out, whereas without QIAAMU it remained pretty constant. This is also to be expected. As various QIAAMU data structures are created and filled by runtime assessment and tradeoff management mechanism, the memory consumption grows. But the runtime memories of QIAAMU processes are not allowed to grow unbounded. After a while old information is discarded; or aggregated and processed information is stored off in databases, which is the reason for the curve flattening out.

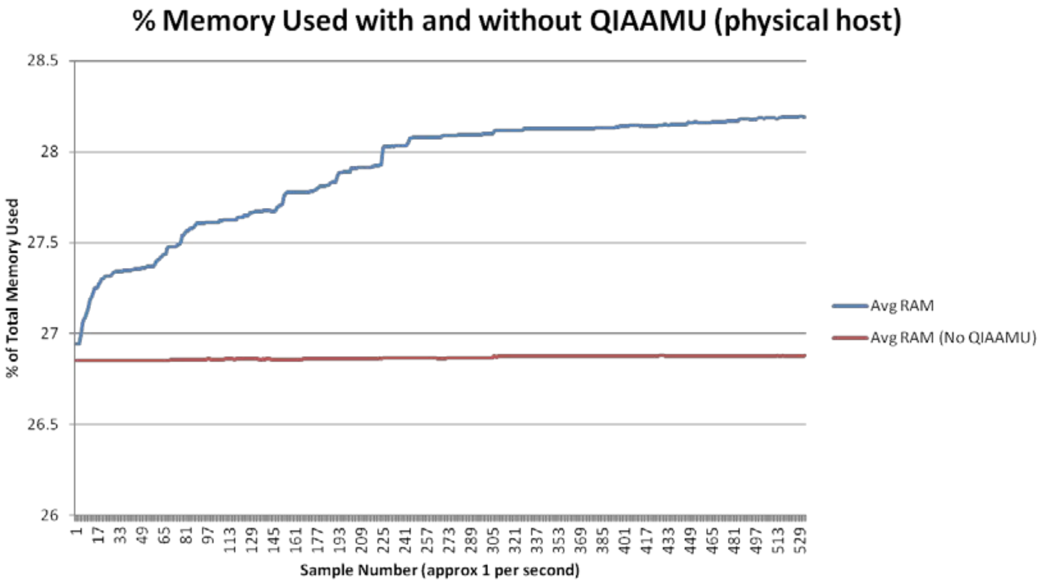


Figure 41: % Memory used with and without QIAAMU for the physical host

In the current setup QIAAMU generates approximately 2-3 web service messages per-second per-blackboard when the system is under active load (indicating trade-offs are potentially occurring with some regularity) as represented by our simulation scenarios. The message size can vary, but a typical message is approximately 500 bytes leading to approximately 1 – 1.5 kilobytes per second added by each blackboard for inter-blackboard/dispatcher communication.

The log messages that are sent to feed the web-based runtime monitor and the ASC-UDOP database generate approximately 11 kilobytes per second per blackboard/dispatcher. These values as is, are quite manageable for an enterprise environment, but could add strain to a constrained tactical network.

4.7 Case Study with a Specific Metric Class (University of Illinois Sub-Contract)

4.7.1 Overview and Short Summary

Part of the QIAAMU solution system is to observe and measure security attributes based on output from sensors such as intrusion detection systems (IDSes). Measuring security is hard and requires developing a rigorous science of security based on sound and reproducible empirical evaluations. This report documents a specific section of the QIAAMU project that consists in exploring the design and development of an accurate and practical security sensor: an anomaly-based intrusion detection system to identify account compromises in a large organization. The objective of such a study is to show how the meticulous assessment of a critical issue combined with best-practice software specifications, development, and testing can lead to a viable security sensor that feeds online security state information to the QIAAMU system.

This focused study started after interviewing security operators at the University of Illinois and at the University of Michigan to identify a critical security issue for which no efficient sensor existed. An emerging problem that offered a particular challenge for the security teams at both institutions was the identification and mitigation of account compromises. Today's connected world is heavily relying on authentication and breaches in the way individuals authenticate can lead to important financial and reputational costs. The number of accounts becoming compromised at both institutions has been increasing over the past few years due to new motivations such as unfettered access to scientific publications and uncensored Internet connectivity, two resources that are free and easily available at academic institutions, given the correct set of credentials. This problem is challenging for security operators because, first, the academic environment is bound to unrestricted Internet access and cannot be limited by stringent security policies. Second, the number of users and the volume of authentication activity are large, giving attackers more opportunities to hide in the noise. Third, both universities lack IT resources to spend time and resources on conducting in-depth investigations of account activity. As a result, the compromise of university accounts has become an attractive target and often leads to accounts being used for illegitimate reasons weeks before they are detected. Most of the time, detection occurs thanks to alerts sent by third parties (e.g., journal publishers receiving too many paper download requests in a short period of time) and lead to loss of reputation for the university as well as potential downtimes for legitimate users.

The approach followed to tackle this issue consisted of three phases. First, we thoroughly investigated the problem by validating a set of hypothesis about victims of account compromise through rigorous statistical testing. Second, we used the understanding gained about the problem to specify, design, and evaluate a machine-learning algorithm that can identify suspicious authentication activity with high accuracy and a low number of false positives. Third, we implemented this algorithm in a production-ready prototype now deployed at the University of Illinois and delivered to AFRL.

This effort led to the development of an efficient anomaly-based intrusion detection system that can identify account compromise in large organizations with high accuracy. The system only requires authentication logs and runs passively every night to output a list of accounts for which suspicious activity was recently detected. The accuracy of the system has been evaluated at two large academic institutions over several weeks and led to the successful detection of unknown compromised accounts with less than two false positive per day.

In addition to the software implementation of the system, an important contribution of this effort has been to explore the design and efficiency of a variety of data mining features for which suspicious activity would likely manifest. As a result, the architecture of the anomaly-based IDS is flexible and can easily adapt to a new set of features, new environments, or even new attack processes.

Finally, this work has contributed to define a rigorous approach to build and evaluate security metrics with respect to the issue of account compromise. This approach covers the data collection, definition of hypothesis, and selection of appropriate statistical tests to understand the key characteristic of the security issue under study. A consequence of applying this approach on the problem of account compromise has been to choose the development of an anomaly-based IDS as an adequate mitigation solution.

This section is organized in four sections. First, a scientific exploration of the security performance of security measures is presented in Section 4.7.2. The results of this study are used to select features and to design the architecture of a machine-learning algorithm to detect account compromises, described in Section 4.7.3. This algorithm has been implemented in a production-ready prototype for which the software documentation is provided in Section 4.7.4. Finally, next steps on the preliminary research of automated responses to extend the IDS into a response and recovery engine are outlined in Section 4.7.5.

4.7.2 Measuring Information Security Performances

Institutions and their users are the target of a variety of threats including malware, botnets, Distributed Denial of Service (DDoS), identity theft, and spam, which challenge the availability of networks and the confidentiality of users' data. To understand how those threats succeed in impacting the security of institutions and how they can be mitigated, researchers must identify and measure the factors that influence both attackers and victims. Quantifying security is particularly important if we are to move from incremental improvements that simply keep pace with those evolving threats to structural or fundamental changes to the security landscape.

While a broad range of security analyses have been attempted—including analyses of what types of rogue software are installed on compromised machines [17], what data are collected by attackers [20, 33], what data are sold [19], and how exploits lead to financial gains [22]—there remains an important gap between the results collected and the development of meaningful security measurements that support the decisions that need to be made to protect systems and networks.

This section presents our attempt to bridge this gap in the context of account compromise at academic institutions. Research results from this work have been described in details in [36]. Our goal has been to empirically evaluate common perceptions about the factors influencing account compromise victims. We translated those perceptions into hypotheses and conducted a study of account compromise incidents at two large university environments over multiple years to confirm or deny our hypotheses. The focuses of our hypotheses are on victim information (e.g., demographics, location, and behavior) and on the effectiveness of several attempts to implement proactive controls (e.g., policy, education) over the security landscape at our institutions. Academic institutions are particularly interesting for this type of study because they offer a significant degree of visibility into both their unique threat landscape (i.e., targeted attacks) and the vulnerability surfaces of their infrastructure. Moreover, they can implement, on reasonably short timescales, proactive and reactive measures to improve their security posture.

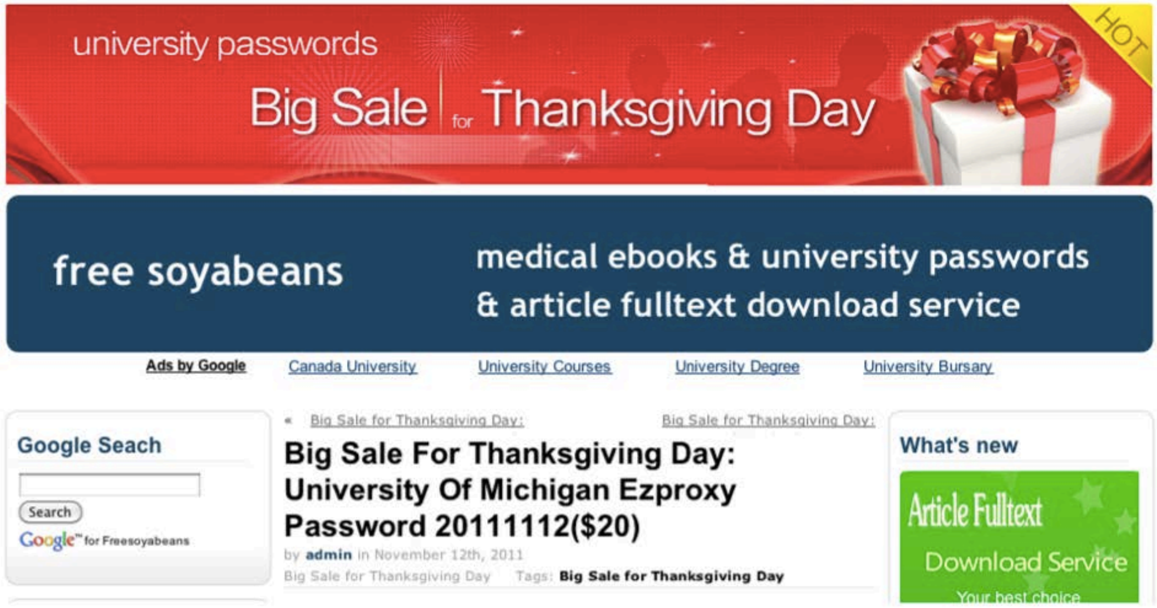
While our study offers several interesting preliminary positive findings (e.g., education level, security training, and network location were significant factors in the susceptibility of victims) and negative findings (e.g., neither gender nor geographic location impacts susceptibility), we make no claim that these results generalize beyond our institutions, nor that we have exhaustively explored or proven any properties about the organizational perspective. Rather, our goal is to highlight this perspective and carefully explain our empirical methodology to encourage other organizations and the community at large to pursue a more quantified approach to security analysis.

4.7.2.1 Background

Starting in late 2010, we engaged in a pilot project with the Office of Information and Infrastructure Assurance (IIA) at the University of Michigan (UofM) and the security team at the University of Illinois at Urbana-Champaign (UIUC) to investigate metrics for enterprise security. Here our goal was similar to that of many existing security metrics efforts:

"IT security metrics provide a practical approach to measuring information security. Evaluating security at the system level, IT security metrics are tools that facilitate decision making and accountability through collection, analysis, and reporting of relevant performance data. Based on IT security performance goals and objectives, IT security metrics are quantifiable, feasible to measure, and repeatable. They provide relevant trends over time and are useful in tracking performance and directing resources to initiate performance improvement actions." [24]

As a first step in this process, the security teams identified account compromise as a pressing security problem, as well as an area in which existing data could be analyzed to build useful metrics.



Big Sale For Thanksgiving Day: University Of Michigan Ezproxy Password 20111112($20)

by admin in November 12th, 2011

Big Sale for Thanksgiving Day Tags: Big Sale for Thanksgiving Day

Figure 42: Online reselling of university credentials

Most universities use basic ID and password authentication methods across systems; therefore, once account credentials are discovered by an illegitimate entity (a person or an automated agent), the account becomes fully compromised.

Such compromised accounts are useful for a wide variety of reasons. Some traditional exploits include resource abuse, accessing of confidential information, spamming, and further credential harvesting. More interestingly, we find that attackers also seek monetary gains by reselling the stolen credentials (as shown in Figure 42). As a motivation for our work on VPN abuse detection [36], an in-depth analysis of the malicious activities revealed that the motivation of purchasers of these credentials is to gain free and unfettered access to information. We showed the stolen credentials were used primarily to download scholarly publications [35] and circumvent state-sponsored censorship (e.g., in China and Iran).

Others have shown that adversaries usually steal university account credentials by attacking authentication mechanisms (e.g., by guessing passwords, exploiting vulnerabilities, or installing trojans), phishing, or social engineering [30]. To mitigate those threats, both *proactive methods* and *reactive procedures* are deployed. While reactive procedures, including compromised account detection and password resetting, are aimed at minimizing the damage after compromises happen, proactive methods are used to reduce the likelihood that end users' accounts become compromised. The proactive methods adopted at universities are mostly part of *education* and *policy* efforts. Education includes publication of online materials and offering of workshops and security quizzes. Preventive policies include password creation and update policies, as well as the implementation of automated account locking after several unsuccessful login attempts.

4.7.2.2 Data used in this study

During 2009, 2010, and the first six months of 2011, IIA recorded 6,600 abuse-related trouble tickets; 1,200 of these were security incidents. The tickets contain information including ticket creation time, category of incidents, comments of security operators, and victim responses. Those tickets are "self-reported" in that the related incidents come directly from university departments, organizations, and service groups, as well as UofM students, staff, and alumni. The tickets cover a wide range of incidents, including "unauthorized exposure of private personal information, computer break-ins and other unauthorized use of UofM systems or data, unauthorized changes to computers or software, equipment theft or loss, and interference with the intended use of information technology resource" [15]. By far the most dominant group of tickets involved unauthorized use of UofM systems, with 822 incidents confirmed (via IIA staff) over two and a half years. At UIUC the number of tickets related to malicious account activity was 178 in 2011. In addition to the trouble tickets, data used in our study include the following:

- *Authentication logs:* The authentication systems deployed at UofM and UIUC share a similar structure. They are both based on Kerberos with different authentication portals (e.g., Web-based services, wireless, VPN). The authentication logs provide useful timing, location, and service usage information, which are critical for tracking user activities.

- *Victim information:* Victim demographics include gender, age, role, nationality, appointment, education level, and working address.

- *Effectiveness of some mitigation strategies:* Records of previous actions that aimed to prevent user account compromise were collected. For example, these include the results of scanning for weak passwords and the list of users who passed a computer security quiz.

A brief note on the sensitive nature of the data: the main risk of this analysis, as identified by the researchers, is informational risk—that is, the psychological, legal, social, and economic damage resulting from inappropriate use or disclosure of information. While the security teams at both institutions oversaw compliance with university practice for the projects, additional steps were taken to mitigate these harms, including assurances that access was for valid statistical purposes, that the researchers used the data appropriately and followed procedures, in some cases that the data were made inherently non-disclosive of any sensitive information, that technical controls surrounding access prevent the unauthorized removal of data, and that the results produced did not contain any protected information [27].

4.7.2.3 Demographic factors

Table 10: Datasets for demographic analysis

University/Year		Total Population	Victims	Victim/Total
UofM	2009	75,992	136	0.18%
	2010	76,944	279	0.36%
	2011	74,346	222	0.30%
UIUC	2011	54,612	178	0.32%

Table 10 shows the data sets used in our demographic analysis. At UofM, IIA collected demographic data for the total on-campus population from January 2009 to July 2011. Over this period, there were 637 victims of account compromises, excluding alumni and former employees. Similar demographic data were collected at UIUC for the year of 2011, and a total of 178 incidents were recorded during this period. We separated accounts into two populations: students and faculty/staff. Within each group, we considered a variety of factors, including gender, age, education level, and citizenship. Table 11 lists this breakdown and the associated subcategories.

Table 11: Demographic variables explored

Group	Variable	Type	Details
Student	Gender	Binary	Male, Female
	Age	Categorical	<19, 20-21, 22-23, 24-25, 26-30, 31-35, >35
	Education	Categorical	Undergraduate, Graduate, Others
	Citizenship	Binary	U.S. Citizen, Non-U.S. Citizen
	Department	Categorical	
Faculty/ Staff	Gender	Binary	Male, Female
	Age	Categorical	<30, 30-39, 40-49, 50-59, 60-64, ≥65
	Education	Categorical	2-year degree, Bachelor (equal), Masters', PhD and above, Specialist, Others
	Citizenship	Binary	U.S. Citizen Non-U.S. Citizen

Methodology: In our analysis, we use *logistic regression*, a form of statistical multiple regression models [25], in order to understand and explain the relationship between multiple user demographic factors and the users' susceptibility to compromises. The multiple regressions allow us to predict the possibility of compromise based on one demographic factor, *holding other factors constant*. For example, this technique is useful to test whether undergraduate students are more likely to become victims than graduate students who are of the same gender, age, citizenship, and department. Thus, we can determine which demographics are predictors of compromise.

The estimated regression model is:

$$L = a+\sum B_i X_i.$$

L in the equation represents the natural logarithm of the odds that the event represented by the dependent variable, which in our case is the account compromise, happens. When p represents the estimated probability that the event happens,

$$L = \ln(p/(1-p)$$

For each predicting variable X_i, the coefficient B_i indicates the amount of change in the natural logarithm with one unit of change in the predictor variable, adjusting for the other variables. For each predictor variable, we test the *null hypothesis* $H_o : B_i = 0$ against the alternative $H_a : B_i \neq 0$. If the null hypothesis is valid, we can conclude that there is not sufficient evidence to indicate that variable X_i is significant in predicting susceptibility, *holding other variables constant*. We use *p-value* as the test statistic and test at a 95% confidence level ($\alpha = 0.05$). The predicting variables with $p - value < \alpha$ are considered significant.

In addition, we recognize that there might be a multicollinearity problem (i.e., correlation among different predictor variables), which could create the false impression that a predictor was significant [25]. Therefore, we use the *Variance Inflation Factor (VIF)* as the diagnostic statistic for multicollinearity. The cutoff value of VIF for determining the presence of multicollinearity is usually 10, which means that values of VIF exceeding 10 are regarded as indicating multicollinearity [18]. By applying this diagnostic on our dataset, we found there is no multicollinearity issue.

Comparing Students with Faculty/Staff One of the questions we wished to explore was *(Q1): which subpopulation is more likely to be infected?* The proportion of faculty and staff members in the total population and in the victim population is shown in Table 12. We note that there are more faculty and staff members at UofM, because employees of the university's hospital are part of our dataset. It can be seen that faculty and staff members show a higher proportion of victims than students do at both universities. In the statistical analysis, the role of faculty/staff also impacts the susceptibility significantly, with p-values of 0.0017 and 0.0006 at UofM and UIUC respectively. In contradiction to our initial belief, members of the faculty/staff population are more likely to become victims than members of the student population.

Analysis on the Student Group: or the student group, we applied logistic regression on all the datasets, and once again we extracted only significant factors using a *p-value* below 0.05. Here we were interested in: *what roles gender (Q2), age (Q3), education-level (Q4), citizenship (Q5), and department (Q6) play in the compromise of student accounts?*

Table 12: Proportions of faculty and staff members in the total population and in the victim population

University/Year		% of total pop.	% of victims
UofM	**2009**	45.16%	57.35%
	2010	45.51%	62.36%
	2011	46.28%	65.17%
UIUC	**2011**	19.68%	26.97%

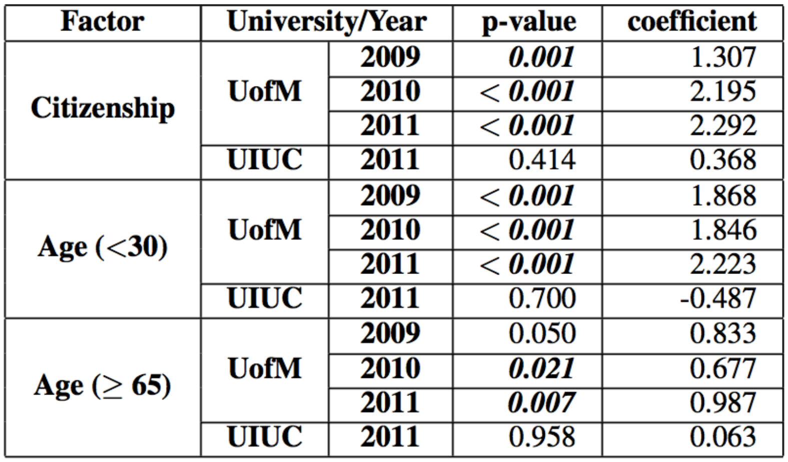
Table 13: Significant influential variables in students

Factor	University/Year		p-value	coefficient
Undergraduate	**UofM**	**2009**	***0.009***	2.957
		2010	<***0.001***	3.520
		2011	***0.020***	3.489
	UIUC	**2011**	0.958	-10.733
Age (20-21)	**UofM**	**2009**	***0.002***	1.219
		2010	***0.004***	0.823
		2011	***0.017***	0.896
	UIUC	**2011**	0.410	-0.472
Citizenship	**UofM**	**2009**	0.520	0.315
		2010	0.659	-0.126
		2011	0.128	-0.460
	UIUC	**2011**	***0.007***	0.5433

Table 13 shows that *education-level* and *age between 20 and 21 years* appear as significant factors influencing susceptibility at UofM. Undergraduate students, who represent 64.78% of the total student population, constituted 87.36% of the student victims in 2011. With p-values lower than 0.05, undergraduate students appear more likely to be compromised than graduate students. Students 20 to 21 years old are the most susceptible to becoming compromised at UofM. About 28.5% of the total student population falls into this age group, but they accounted for 38.89%, 37.14%, and 47.13% in the student victim population in 2009, 2010, and 2011 (with p-values of 0.002, 0.004, and 0.017). However, there is insufficient evidence to show that those two factors are also significant at UIUC.

Citizenship appears as a significant predictor of susceptibility for UIUC but not for UofM. In 2011, foreign students accounted for about 22% of total students at UIUC. Meanwhile, 33.58% of the student victims were foreign, indicating that foreign students were more susceptible to compromise than domestic students were. However, similar results were not found at UofM. In addition, we found that *gender*, which has a p-value of 0.07, is potentially a useful predicting variable at UIUC, where males are more susceptible than females. The data shows that 64.92% of the student victims were male, while only 53.84% of the total student population was male.

Table 14: Significant influential variables for faculty/staff



Factor	University/Year		p-value	coefficient
Citizenship	**UofM**	**2009**	***0.001***	1.307
		2010	< ***0.001***	2.195
		2011	< ***0.001***	2.292
	UIUC	**2011**	0.414	0.368
Age (<30)	**UofM**	**2009**	< ***0.001***	1.868
		2010	< ***0.001***	1.846
		2011	< ***0.001***	2.223
	UIUC	**2011**	0.700	-0.487
Age (≥ 65)	**UofM**	**2009**	0.050	0.833
		2010	***0.021***	0.677
		2011	***0.007***	0.987
	UIUC	**2011**	0.958	0.063

Analysis on the Faculty/Staff Group: We performed a similar analysis on the faculty/staff group. *What roles do gender (Q7), age (Q8), education (Q9), and citizenship (Q10) play in account compromises?* Table 14 shows the significant factors that influence victims among faculty and staff members. Results reveal that *citizenship* and *age* are significant in predicting user susceptibility to account compromise at UofM. Foreign employees are more likely to become victims than U.S. citizens are. While about 2.1% of employees at UofM are not U.S. citizens, they account for 8%, 17.18%, and 18.87% of the faculty/staff victims in 2009, 2010, and 2011 respectively (with p-values lower than or equal to .001). Although we observed more foreign faculty and staff members than domestic employees compromised, the null hypothesis test shows insufficient evidence to conclude that citizenship is a significant predictor at UIUC.

4.7.2.4 Temporal factors

We are also interested in understanding when account compromises occur. In particular, we wish to know *(Q11): whether the incidence of compromises varies at different times of the year.* We performed a time series data analysis on the monthly number of tickets at UofM via the "Holt-Winters" exponential smoothing procedure [34]. The analysis decomposes the time series data into *long-term trend* and *seasonality*, and while the long-term trend shows the number of incidents increasing, the fit to the seasonality is poor. As shown in Figure 43, the predicted seasonal effects are absent from the real-world observation. Also, an issue to notice when evaluating such temporal effects is that the timestamps linked to tickets represent ticket creation time rather than compromise time. As a result, they may be too coarse-grained for a monthly frequency analysis.

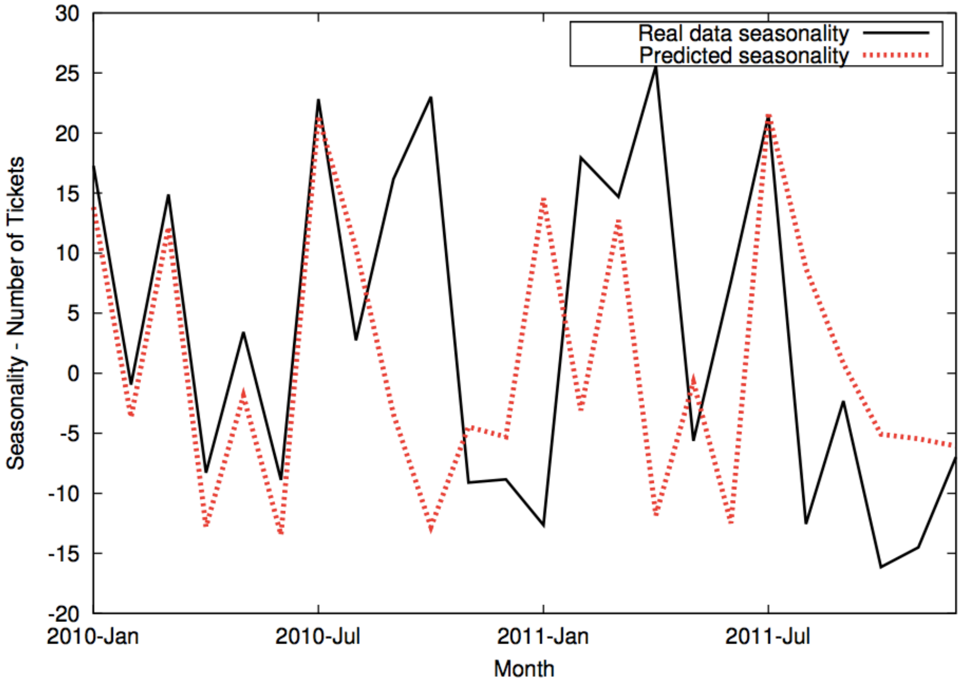


Figure 43: Seasonality in the number of tickets at UofM

4.7.2.5 Geographical factors

In this analysis, we attempted to determine *(Q12): whether victims are clustered geographically*. For example, are specific buildings or campuses more susceptible than others? We hypothesized that attackers taking advantage of user behavior or targeting shared infrastructure will likely have an impact on a victim's susceptibility. Figure 44 shows the breakdown of victim university locations by university mail code at UofM. Mail codes offer a coarse-grained geographic measure to group university buildings. As one can see, nearly 70% of mail codes contained only one victim, with no mail code containing more than eight victims over the two-year period.

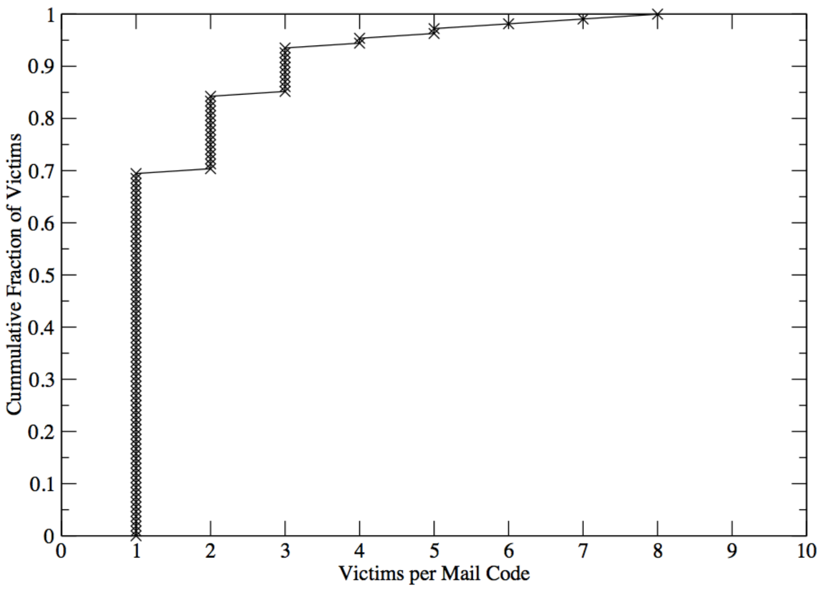


Figure 44: Cumulative fraction of victims per location

4.7.2.6 Topological factors

While the above analysis showed no strong geographic bias, we further hypothesized that infrastructure vulnerabilities could be an important reason for compromises. Here we modify our question: *(Q13) Are victims clustered topologically?* Rather than physical location, we are concerned with virtual location in the network topology. Figure 45 shows the result of victim location, as defined by resources used in a given network administrative domain. A handful of networks stand out as clusters of activity. The top five include two computer laboratory domains, the VPN domains, one school building, and the electronic library resources.

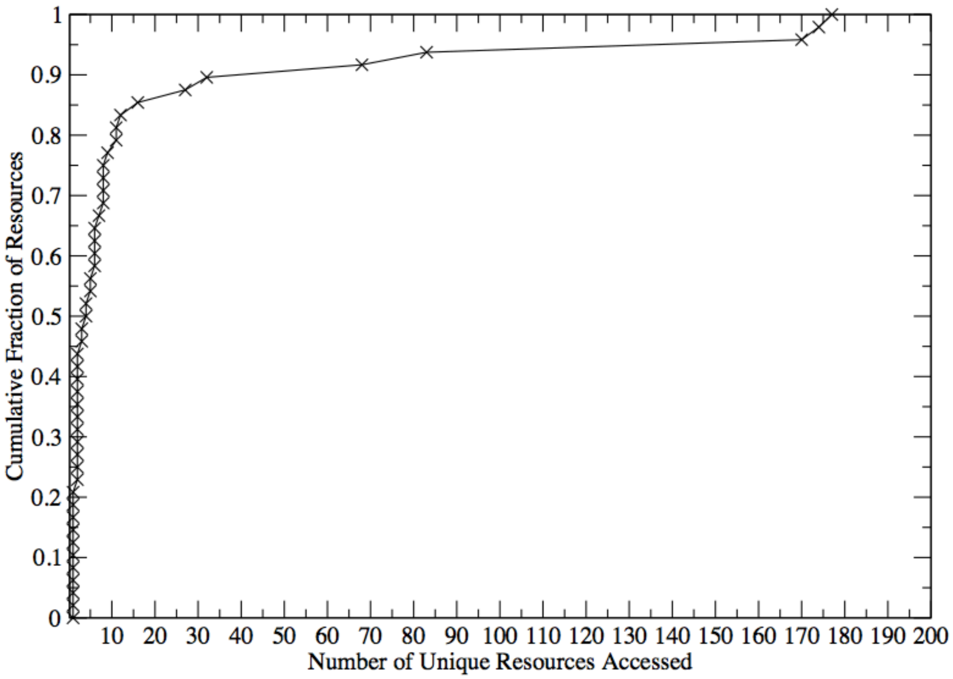


Figure 45: The cumulative number of unique resources used by victims in each administrative domain

4.7.2.7 Service behavior

At both universities, various online services are provided, among which three basic services occupy the majority of user activities: *Web-based services*, *Wireless service* and *VPN service*. Each service authenticates separately, and therefore we can infer usage frequency of different classes of service. We are interested in *(Q14) Do usage patterns of victims, attackers, and benign users vary?*

We evaluate the portal usage model of three different groups of users at UofM: users who never have been reported as behaving maliciously (referred to as *normal users*), victims who were under control of adversaries (referred to as *victims/adversaries*), and previous victims who have been compromised but now claim to be benign (referred as *previous victims*). Since we aim to find whether a compromise is related to specific services, the comparison between *normal users*, *previous victims* and *victims/adversaries* might indicate a good way to design a compromised account detection solution or to enhance a university policy. We calculated the usage percentage of each service based on the number of successful login sessions in a period of one week during February 2011 at UofM. The data includes 113,644 *normal users*, 261 *previous victims* and 121 *victims/adversaries*. We show the complementary cumulative percentage of users by the proportion of their usage of each portal in Figure 46. In examining all three figures, we see that *previous victims* were heavy users of Web services, while *normal users* and *victims/adversaries* had similar profiles. *Victims/adversaries* and *previous victims* shared similarities in wireless usage, while *normal users* were much heavier users of the service. *Normal users* and *previous victims* were nearly non-existent in VPN usage, while *Victims/adversaries* showed heavy usage.

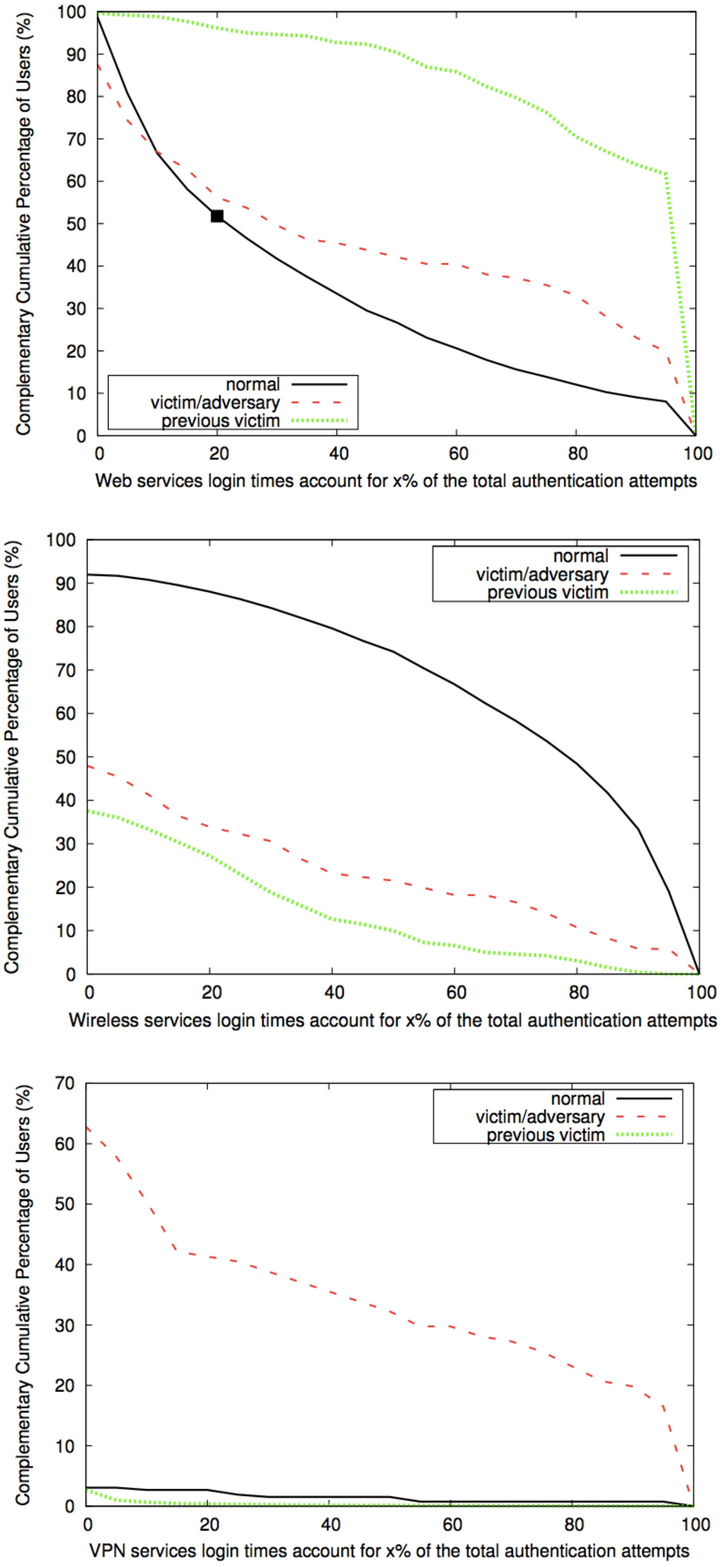


Figure 46: Complementary cumulative percentage of users whose usage of the service is more than *x*% of the total usage for different services

Figure 46 shows complementary cumulative percentage of users whose usage of the service is more than x% of the total usage for three services. If we take the square point in (a) as an example, it indicates that for about 50% of normal users, the web-based service usage accounts for more than 20% of their total service usage (i.e. *number of login sessions to web−based services* > 20% of *total authentication attempts*.) Intuitively, curves to the upper left indicate populations who use this service frequently and curves to the lower right indicate populations of infrequent users.

4.7.2.8 Password strength

The password creation policy aims to prevent the adoption of weak passwords that could be easily cracked. At UofM, the operation team performs weak password scanning every six months, in an effort to reduce the impact of weak passwords. Here, we explore the question: *(Q15) Are accounts with weak passwords more likely to be compromised?*

At UofM, 380 accounts were compromised, while 2,284 accounts with weak passwords were detected in 2012. Only 12 accounts are present in both sets, accounting for 3.16% of compromised accounts and 0.52% of weak password accounts. Given the small intersection, we may infer that while weak passwords do play some role in the account compromises, they are a very minor attack vector.

Table 15: Probability of account compromise for the total population and the weak password accounts

	# of total	# of compromised	Pr (compromise)
Total population	550,000	380	0.069%
Weak password	2,284	12	0.525%

However, as shown in Table 15, the probability of a weak password account being compromised is much higher than that of the total population. Note that the total number of accounts at UofM is 550,000, because alumni accounts can continue to use some university services. In order to validate the significance of this observation, we performed a *test of homogeneity* [16], which is a statistical hypothesis testing method for categorical data. Here, the null hypothesis, Ho, is that accounts with weak passwords have the same probability of compromise as that of other accounts. The alternative hypothesis, Ha, is that weak password accounts have a higher probability of compromise. We would reject Ho if $p - value < 0.5$. In our study, the test result has a deviance of 28.09 and a p-value of 1.16^{-16}. Therefore, we reject Ho and conclude that there is sufficient evidence to show that accounts with weak passwords are more likely to be compromised.

4.7.2.9 Education impact

Since 2005, students at UofM have participated in an annual voluntary online computer security quiz. We were curious to answer *(Q16) Does the security quiz positively impact a user's ability to keep an account secure?* We examined the list of people who passed the quiz (referred to as educated users) and found 17,625 distinct users. Among those users, 23 were victims of account compromise. Figure 47 shows the number of participants by year. The quiz can be taken multiple times, and new users who have not taken the quiz before represent on average 71.94% of the total number of users. 27.53% of the participants took the quiz over multiple years, or several times during a single year.

To understand the effectiveness of education, we compare the probability of falling into compromise for two groups in 2010: the educated user group and the total on-campus student group. The total on-campus student population represents 41,924 students, among which 9,227 have passed the security quiz. There were 105 student accounts that were reported to be compromised in 2010, among which nine were from the educated user population. The probability of educated users becoming compromised was 0.1%, while the total on-campus student population had a probability of 0.25%. Again, we applied a test of homogeneity to determine if this difference is significant. Here, the null hypothesis, Ho, is that all the students passing the security quiz have the same probability to become victims as those that do not, while the alternative, Ha, is that students who pass the quiz are less likely to be compromised. The results of the test show a deviance of 13.52 and a p-value of 2.36^{-4}. Since the p-value is less than 0.05, we can reject Ho and conclude that people who passed the quiz were less likely to become victims.

However, we can only infer that people who took and passed the security quiz had better awareness and knowledge of security. Thus the question that how effective the security quiz is in mitigating the compromise account problem remains unanswered in our work.

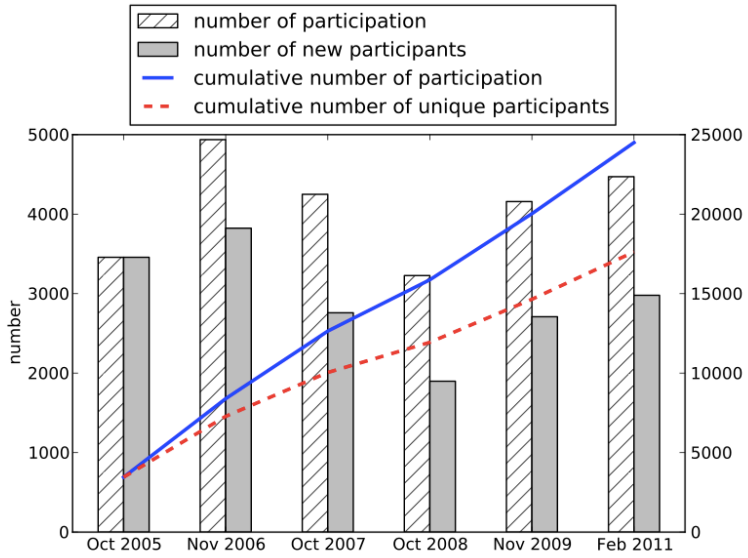


Figure 47: Participation in security quiz

4.7.2.10 Lessons learned

We believe in the need for relevant measurements and metrics as we seek to build a science of security. As a result, one effort of this work has been to think beyond the tactical response to our analysis and to answer more fundamental problems, such as choosing appropriate metrics, creating methods for validating metrics, comparing methods for metric computation and collection, understanding the appropriate uses of metrics (operations, evaluation, risk management, decision making), and building the ability to measure operational security values [29].

From Observations to Hypotheses. A striking lesson from this work is the reminder that human observation, while being the launching point for scientific inquiry, is itself only the beginning of the process. Several of the hypotheses we developed based on our observations proved to be invalid through thorough statistical testing. For example, with regard to Q1, we expected that less aversion to risks and lower levels of experience would make students more susceptible to compromise than the faculty/staff group. With Q2, previous studies had shown that gender played a role in phishing susceptibility [21, 31, 23], and we were interested in determining if this factor is also influential in our datasets. The fact that it either did not matter or that male students were more likely (in the case of UIUC) was unexpected. Answers to Q11 also ran counter to both our expectation and to the security teams' experiences. Observations, such as on-campus student populations or overall traffic volumes, exhibit strong seasonal patterns per semester. The lack of those patterns in our data was surprising. Finally, we hypothesized that Q12 would be an excellent metric, as we felt geographic location encompassed factors such as targeted and shared environments, which could influence subpopulation susceptibility. Our data and analysis did not support this hypothesis.

Publication and Validation. An important challenge in a variety of security domains is the design of reproducible experiments [32, 28, 26]. The results of poor metric selection, the misapplication of scientific analysis techniques, or inadequate technical details greatly diminish the value of empirical results. We experienced this most acutely in the use of trouble tickets as the main source for metrics in our study. Uncertainty in the date of compromise, the source of trouble tickets, or the ticket reporting methodology (e.g., should a large break-in be recorded in a single or multiple ticket?) challenge the validity of our results.

Generalization. In addition to building confidence in the findings of an individual study, cross-validating those findings helps us to understand their generalizability. For example, we found that the lack of agreement across institutions for questions Q2-Q10 was interesting. While we believe differences in UofM and UIUC metrics and evaluations may exist, we hypothesized that these social, behavioral, and economic factors would have a more universal impact on compromises.

Facilitating Decision Making. Metrics are, of course, just the first step in building more secure enterprises. Using metrics to answer important questions and to guide future actions are necessary next steps in any formal security process. In our analysis this was best highlighted in the answers for Q15-Q16, which confirmed the role of education and password strength in account compromise. However, such correlations do not help us make value judgments about the utility of education or password cracking efforts. Are the numbers of potentially impacted accounts worth the effort? How do these efforts compare with other opportunities?

4.7.2.11 Conclusions

In this section, we presented efforts in trying to understand the security of academic enterprise networks. Focusing on university account compromises, we have identified a series of questions to help guide the more general development of security metrics within those networks. We examined questions in the specific context of account compromise incidents at UofM and UIUC over a three-year period. Those questions cover a variety of analysis, including demographic, temporal, geographic, topological, and behavioral factors that may influence compromise, as well as the effectiveness of existing policy and security training efforts at UofM.

As part of future work, we are interested in evaluating our findings as a guide to future proactive mitigation efforts. Such evaluation requires collaborating further with the operational teams to enhance and evaluate proactive initiatives as well as to investigate deeper the modus operandi of attackers (e.g., by using honeypots) and the behavior of victims (e.g., by surveying the campus population). We are also aware that our results remain preliminary and may be specific to our two institutions. Therefore, we are interested in involving additional organizations and extending our analysis to a larger scale.

From a broader point of view, this works fits into a long-term effort by the community to build a science of enterprise security, which provides a quantified and continuous security process [29]. A cornerstone to reach this ambitious goal is to define a framework for interdisciplinary and cross-institutional research on enterprise security. While the challenges are well known and significant (they include data availability, data sharing, privacy issues, as well as efficient solutions to enable collaboration among stakeholders from different backgrounds and from heterogeneous institutions) this work offers a concrete example to guide future efforts toward addressing them.

4.7.3 Safeguarding Academic Accounts and Resources with the University Credential Abuse Auditing System

As shown in the previous section, compromised accounts are an increasing issue for academic organization and stolen credentials are used to actively utilize Virtual Private Network (VPN) and library publication resources. The VPN enables attackers to bypass censorship mechanisms deployed in their countries. Recently, the exploitation of scholarly databases has also become a lucrative business, as attackers download a large number of articles and then resell them in underground markets [31].

To address the problem, we present in this section the design, implementation, and evaluation of a VPN abuse detection system that focuses on supplementing existing security measures to rapidly identify account compromises. For more details, see [36]. Based on lessons learned during the analysis of incidents (see previous section), our system analyzes authentication logs on a daily basis and reports accounts for which suspicious activity is detected. The detection technology is based on a machine-learning approach that automatically generates a set of features before classifying user activity. Our work makes three important contributions. First, we report on the motivation of attackers who compromise academic accounts, based on several years of incident analysis at two large universities. Second, we present the design of an authentication log analysis solution that can process the daily activity of thousands of accounts with high accuracy and a low false-positive rate. Third, we evaluate this system on several weeks of logs at each university in close collaboration with the institutions' security teams. This large-scale experiment has led to interesting insights about the specific challenges of analyzing malicious activity from campus data.

4.7.3.1 University authentication infrastructure

Universities usually provide an abundance of resources and online services to vast and diversified user groups at both the campus and college levels. For example, users have access to webmail, VPN, course registration, online storage, payroll and employment information management, and library resources. According to data collected in November 2010, there are 41,924 students and 34,947 faculty and staff members (including university hospital personnel) at UofM, but the total number of unique accounts reaches 556,281 because alumni and former employees can continue to use their accounts after they leave the campus. Among those accounts, 206,529 had recent activity. At UIUC, the population includes 54,612 people, including 43,862 students and 10,750 faculty and staff members. Unlike UofM, UIUC locks accounts a few months after students graduate or employees leave the university.

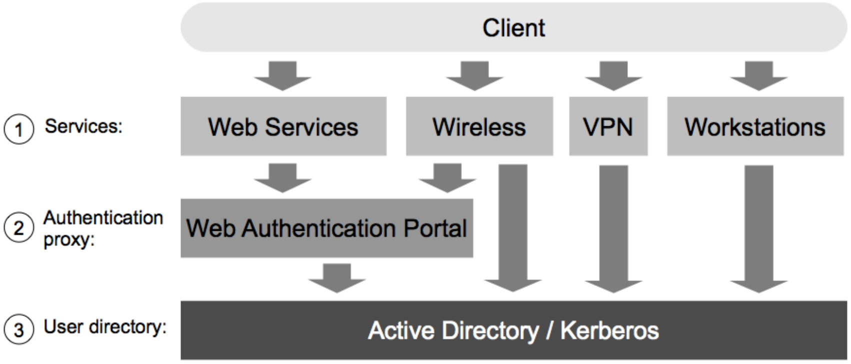


Figure 48: Authentication infrastructure

As shown in Figure 48, UofM and UIUC share a similar authentication infrastructure built around the Kerberos authentication protocol and different authentication portals for various services. One is the single-sign-on protocol for Web services (Cosign at UofM and Bluestem at UIUC). Both Cosign and Bluestem are browser-based Web authentication solutions that enable users to access restricted online resources or Web applications at the universities. The other important authentication channel is the VPN service. Users can gain remote access to the university network through different VPN clients. Logs from all the VPN gateways are recorded. Those two types of services are the main focus because they represent the majority of campus usage and account compromises are likely to manifest in their logs.

4.7.3.2 Overview of detection system

1) Goals: The main design goal of the University Credential Abuse Auditing System (UCAAS) is to assist security teams by automatically flagging compromised accounts. The first requirement is that the system must be effective in detecting compromised accounts, even if the illegitimate activity is stealthy. The effectiveness of UCAAS relies heavily on the account activity features chosen for classification. We engaged in discussions with the security teams at both universities, and we examined authentication logs from the past two and a half years in order to gain a detailed understanding of legitimate and malicious activity. As a result, we carefully selected a combination of features in which illegitimate activity would be most likely to manifest. We describe our selection in Section III-B and evaluate it in Section IV-B. Second, given the relative scarcity of compromised accounts compared to the total population, the distribution of false positives over true positives is likely skewed [37]. The cost of false positives impacts both the operational team, which has to spend time investigating flagged accounts, and legitimate users, because their accounts could be blocked. Discussion with the security teams helped us to define the requirement of a maximum average number of *two false alarms per day*.

2) Overall Design: UCAAS detects suspicious accounts based on authentication logs collected from university systems. Since VPN is used as an entry point by attackers, we first filter authentication activity to keep only logs generated by users who accessed the VPN at least once. The second step consists of extracting and analyzing the set of features for the daily activity of each account. The activity captured by the different features describes the behavioral, geographical, and topological pattern information, as well as possible deviations from historical profile data. Finally, a classifier runs on the feature vectors to determine whether an account is compromised or not. The model used by the classifier is trained and built automatically from the past n days of authentication logs. This training dataset is made of both known compromised and legitimate accounts, and is updated dynamically over time.

4.7.3.3 Features developed

1) Suspicious Behavior Features: First, we created three heuristics based on our analysis of compromised account behavior patterns. The thresholds in the heuristics are transparent, as they were derived from operator experience.

Temporal-Spatial Violation: This feature captures accounts that had activities from geographically different locations in a short period of time. The key insight here is that the activities of legitimate account owners and attackers are independent. Thus, if attackers and account owners are located in different places, they will likely generate inconsistencies in location and time of activities. Since attackers can neither predict nor control the legitimate users, this detection can hardly be evaded.

Suspicious IP Addresses: UCAAS labels an IP address as suspicious if it was used by more than one account to log in during one day. An analysis of historical compromised account authentication logs revealed that a handful of IP addresses scanned multiple compromised accounts within a short period of time. In fact, at UofM, more than half of the compromised accounts reported in 2009 and 2010 were accessed by those malicious IP addresses.

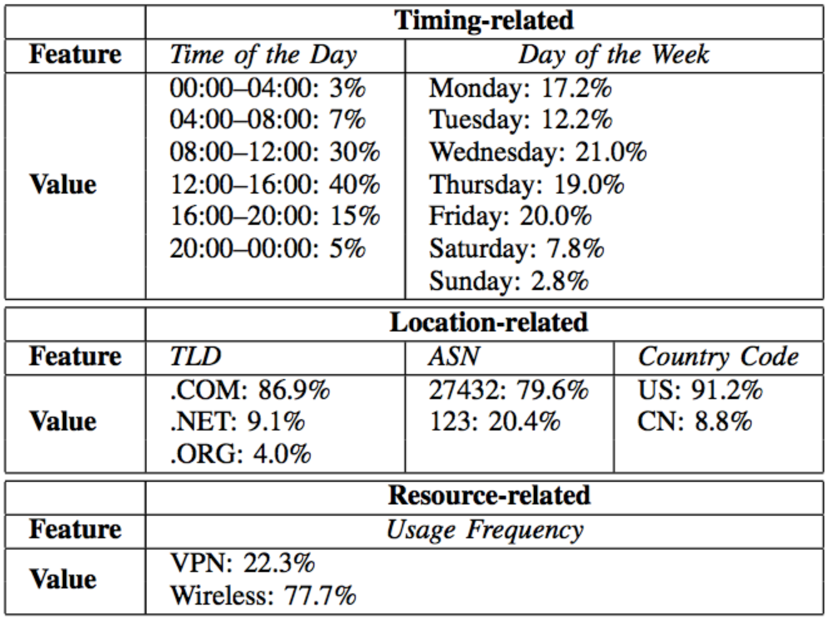
Suspicious Usage Pattern: Account activity is labeled as suspicious if it consists exclusively of a combination of VPN and library accesses. Unlike the first feature, which relied on overlapping legitimate and malicious activities, this feature aims to detect idle accounts that are used only by the compromising entity. For example, alumni and student accounts are mostly inactive during vacation periods; once those accounts have been compromised, their usage patterns no longer reflect traditional academic activities (e.g., registering for classes, connecting to the wireless network). Note that we use a threshold to characterize how exclusive the usage pattern should be before an alert is raised. This is to prevent attackers from evading the heuristic by randomly logging in to resources other than the VPN and the library.

While this first group of features focuses on defining specific signatures for illegitimate activity, the remaining three groups of features take a complementary approach by learning anomaly-based characteristics over time.

2) IP Address-based Features: We observed that the geographical and topological distributions of attackers were not uniform. As a result, UCAAS generates three IP address-based features to capture client origins that are likely linked to illegitimate activity: *geographic location*, *Autonomous System Number (ASN)*, and *Top-Level Domain (TLD)*.

3) Resource Usage-based Features: To expand on the suspicious usage pattern heuristic, the system learns the ratios of resource usage for the account. These ratios are computed for each account based on the numbers of VPN connections and university website accesses. This set of features is based on the intuition that legitimate users usually go to a variety of online services provided by their universities, while attackers only exploit a few services heavily.

Table 16: Sample user profile learned over a week of activity



	Timing-related	
Feature	*Time of the Day*	*Day of the Week*
Value	00:00–04:00: 3% 04:00–08:00: 7% 08:00–12:00: 30% 12:00–16:00: 40% 16:00–20:00: 15% 20:00–00:00: 5%	Monday: 17.2% Tuesday: 12.2% Wednesday: 21.0% Thursday: 19.0% Friday: 20.0% Saturday: 7.8% Sunday: 2.8%

	Location-related		
Feature	*TLD*	*ASN*	*Country Code*
Value	.COM: 86.9% .NET: 9.1% .ORG: 4.0%	27432: 79.6% 123: 20.4%	US: 91.2% CN: 8.8%

	Resource-related
Feature	*Usage Frequency*
Value	VPN: 22.3% Wireless: 77.7%

4) Profile-based Features: Finally, for each account, UCAAS computes the probability that the latest activity recorded matches the historic profile of user activity collected over the past week. An example profile is presented in Table 16. If we observe a new authentication attempt from the United States for this sample account, the probability that this attempt fits in the historical usage pattern is 91.2%. The idea behind these features is that illegitimate activity will not match the authentication habits of account owners.

4.7.3.4 Classification

The next step after computing values for the different features is to classify accounts into two classes: benign or suspicious. For this task, UCAAS uses a logistic regression classifier implemented in Weka [38]. We initially ran experiments with a variety of machine-learning algorithms, including support vector machine, naive Bayes, and K-nearest neighbors algorithms. Most of them offered good results if the parameters were correctly tuned. Overall, we found that logistic regression classifier provided the best accuracy.

Indeed, a logistic regression model [25] is inherently suitable for single dichotomous label classification. The regression model to calculate L, the odds that a compromise happens, is similar to the model presented in subsection 2.3. The final classification uses a threshold to identify accounts as benign or possibly compromised.

4.7.3.5 Evaluation Datasets and Ground Truth

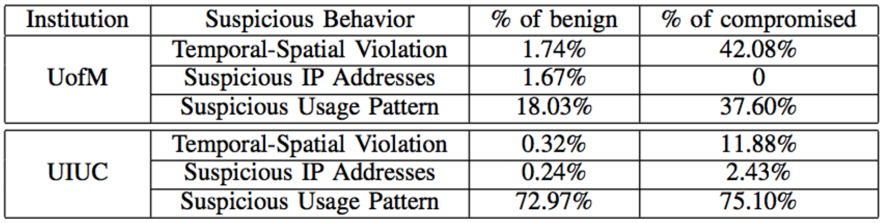
We used two datasets to evaluate UCAAS: a *training set* used for feature tuning and model testing, and a *validation set*. One critical and difficult step in building the training set is that of obtaining the ground truth. We addressed this challenge through a close collaboration with the security teams, which have acquired extensive experience in dealing with compromised accounts over the past several years. The first step was to collect known incident tickets from 2009 to 2011. They represent a subset of the total compromised accounts, since many compromised accounts are never identified. Therefore, we needed further manual checking of the dataset. However, the large volume of logs in our dataset (around 6 million) made the manual validation of each authentication attempt impractical. To address this problem, we ran the heuristics discussed in Section III-B1 with conservative parameters tuned to minimize false negatives. The set of flagged accounts was sent to the security team. They examined each account and contacted the owners of suspicious ones via email and telephone to assess whether or not the suspicious activity detected was really illegitimate. Most of the time, this validation step consisted of asking if the user traveled to the foreign country identified in the data, or if the user shared credentials with other people. For alumni or previous employees at UofM who could not be reached by email or telephone, the security team provided an expert judgment. The final step was to manually examine the authentication activities of those compromised accounts, and label their feature vectors as compromised in the days during which we had high confidence that illegitimate activity had occurred. We kept refining the ground truth with the latest detection results during the course of the evaluation process. We believe that through this process, most of the accounts were correctly labeled. It should be noted that users who shared their credentials with their families and friends living abroad were discarded from our dataset. The reason was that those accounts were not compromised, although their activity matched that of compromised accounts. The accuracy of the classifier would be negatively impacted if those accounts were labeled as benign. In addition, although detection of shared credential accounts is not a goal of UCAAS, we did not count them as false positives if they were flagged by UCAAS, because sharing of credentials is discouraged by the universities.

At UofM, the training data were collected from June 14 to June 28, 2011. The dataset includes 108,366 unique users who had 2,129,275 authentication attempts. After filtering out users who did not have VPN-related activities and conducting the validation process, we got a final training set of 2,441 benign and 87 compromised accounts. The empirical evaluation was done on a different validation set collected from September 14 to October 2, 2011 and consisting of 6,562,153 login sessions from 127,316 unique users. At UIUC, the training data were collected from June 19 to July 2, 2011. There are 104,172 successful logins from 25,530 users in the dataset. After the filtering and validation, we got a final training set of 4,692 benign and 6 compromised accounts. The evaluation set was collected from July 9 to July 23, 2011 and consists of 106,477 logins from 24,979 unique users. The limited number of compromised accounts at UIUC led us to add 10 incidents to the set. Those incidents were detected by the security team during the first half of 2011. We carefully examined the impact of adding those incidents to make sure that they would help to improve model accuracy without affecting the false positive rate.

4.7.3.6 Feature Evaluation

In this subsection, we analyze the effectiveness of the feature set by comparing the proportions of benign and compromised accounts that were flagged by each feature. We also list the coefficients for significant features, as calculated by Weka when building the model. As explained previously, the coefficients represent the contribution of each feature to the model, so the presence of features with high coefficients indicates accounts that were likely compromised.

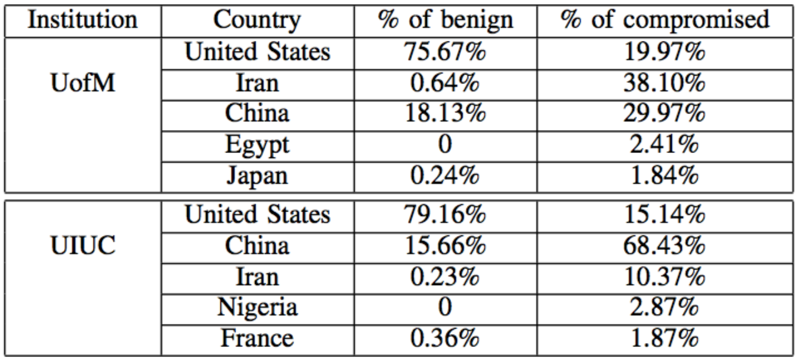
Table 17: Proportions of authentication attempts from benign accounts and compromised accounts flagged as suspicious



Institution	Suspicious Behavior	% of benign	% of compromised
UofM	Temporal-Spatial Violation	1.74%	42.08%
	Suspicious IP Addresses	1.67%	0
	Suspicious Usage Pattern	18.03%	37.60%
UIUC	Temporal-Spatial Violation	0.32%	11.88%
	Suspicious IP Addresses	0.24%	2.43%
	Suspicious Usage Pattern	72.97%	75.10%

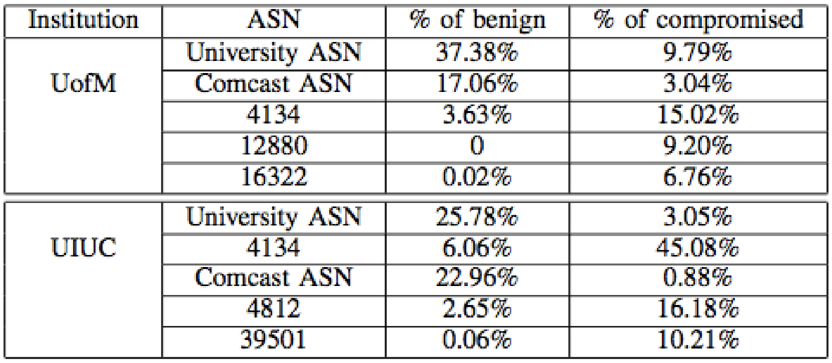
The results for the set of suspicious behavior features are shown in Table 17. We find that for both institutions, a higher proportion of compromised accounts manifest suspicious behavior. The only exception was that, interestingly, no compromised account was reported as having a suspicious IP address in the training set at UofM. This indicates a drastic evolution of the threat model, since half of the incidents reported in 2010 were linked to suspicious IP addresses. The coefficients for the feature temporal-spatial violation, suspicious IP addresses, and suspicious usage pattern at UofM are 6.81, 8.11 and -35.39 (sign of a coefficient indicates the direction of influence), respectively. It is in accordance with our observations that in this model, suspicious IP addresses are highly negatively correlated with compromises while the other two suspicious behaviors are positive indicators of compromises.

Table 18: Distribution of IP address geolocation



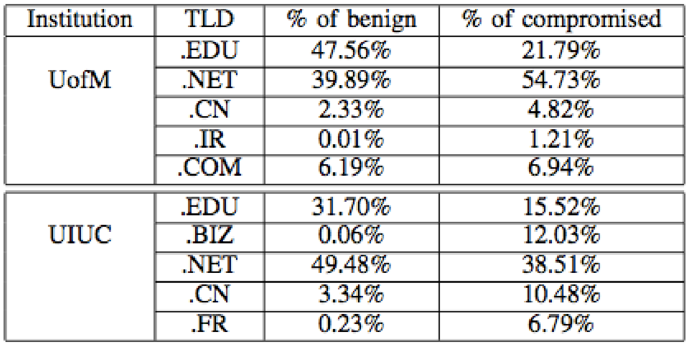
Institution	Country	% of benign	% of compromised
UofM	United States	75.67%	19.97%
	Iran	0.64%	38.10%
	China	18.13%	29.97%
	Egypt	0	2.41%
	Japan	0.24%	1.84%
UIUC	United States	79.16%	15.14%
	China	15.66%	68.43%
	Iran	0.23%	10.37%
	Nigeria	0	2.87%
	France	0.36%	1.87%

Table 19: Distribution of ASN



Institution	ASN	% of benign	% of compromised
UofM	University ASN	37.38%	9.79%
	Comcast ASN	17.06%	3.04%
	4134	3.63%	15.02%
	12880	0	9.20%
	16322	0.02%	6.76%
UIUC	University ASN	25.78%	3.05%
	4134	6.06%	45.08%
	Comcast ASN	22.96%	0.88%
	4812	2.65%	16.18%
	39501	0.06%	10.21%

Table 20: Distribution of TLDs

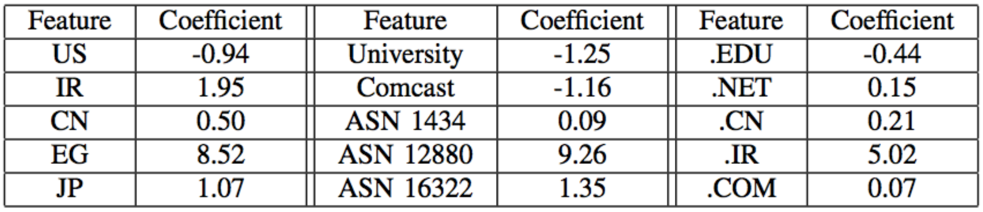


Institution	TLD	% of benign	% of compromised
UofM	.EDU	47.56%	21.79%
	.NET	39.89%	54.73%
	.CN	2.33%	4.82%
	.IR	0.01%	1.21%
	.COM	6.19%	6.94%
UIUC	.EDU	31.70%	15.52%
	.BIZ	0.06%	12.03%
	.NET	49.48%	38.51%
	.CN	3.34%	10.48%
	.FR	0.23%	6.79%

We then analyzed the performance of IP-based features. The top five country codes, ASNs, and TLDs are shown in Table 18, Table 19 and Table 20, respectively. It is interesting to observe the similarities between the two institutions. For instance, the large majority of legitimate activity occurs in the United States and within the borders of the universities, and the top two countries linked to illegitimate activity are similar, along with an ASN of China-Telecom (4134). On the other hand, the rankings of TLDs and ASNs also reveal some differences. Although the training datasets were collected around the same time, we observed that illegitimate activity came from different sets of internet service providers for the two universities.

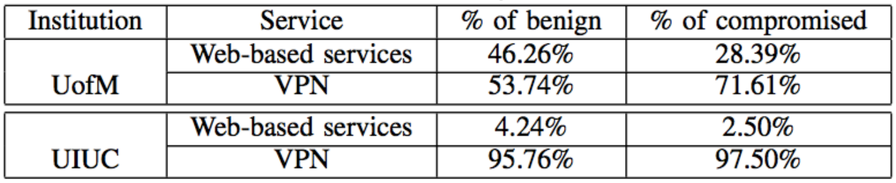
When looking at the discriminating coefficients of some of those features at UofM shown in Table 21, we get the same results. The features of country being China or Iran, ASN being 4134, 12880, 16322, and TLD being .NET, .CN, or .IR all have positive coefficients, while the United States, the University's ASNs, and the TLD .EDU have negative coefficients.

Table 21: Discriminating coefficients of IP-based features at UofM



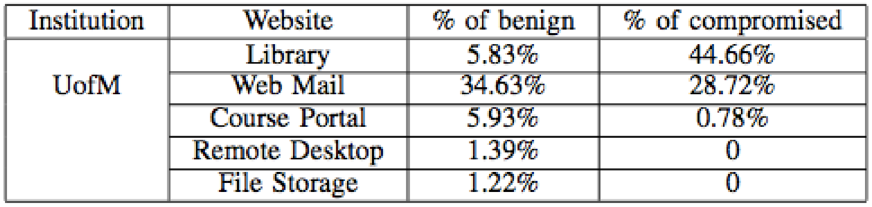
Feature	Coefficient	Feature	Coefficient	Feature	Coefficient
US	-0.94	University	-1.25	.EDU	-0.44
IR	1.95	Comcast	-1.16	.NET	0.15
CN	0.50	ASN 1434	0.09	.CN	0.21
EG	8.52	ASN 12880	9.26	.IR	5.02
JP	1.07	ASN 16322	1.35	.COM	0.07

Table 22: Service usage distribution

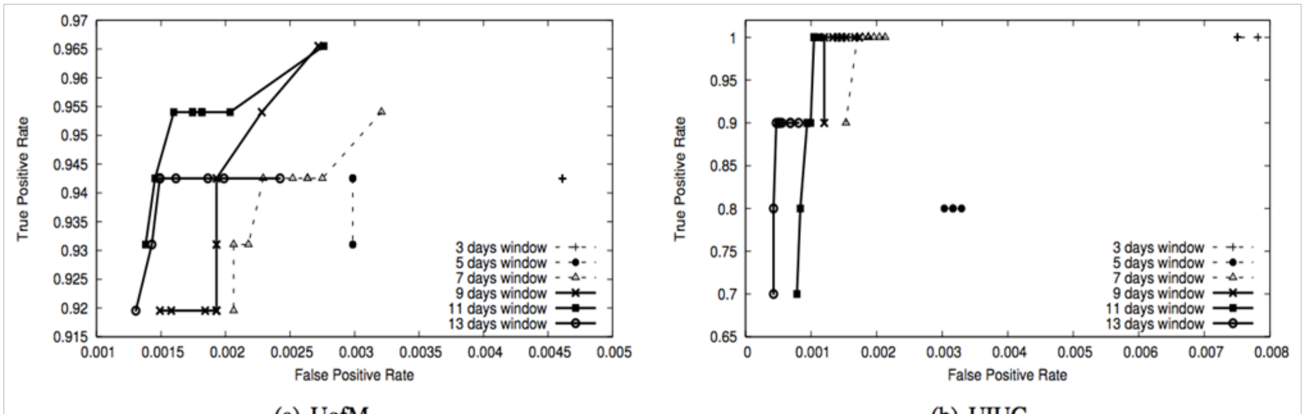


Institution	Service	% of benign	% of compromised
UofM	Web-based services	46.26%	28.39%
	VPN	53.74%	71.61%
UIUC	Web-based services	4.24%	2.50%
	VPN	95.76%	97.50%

Table 23: Distribution of websites visited at UofM

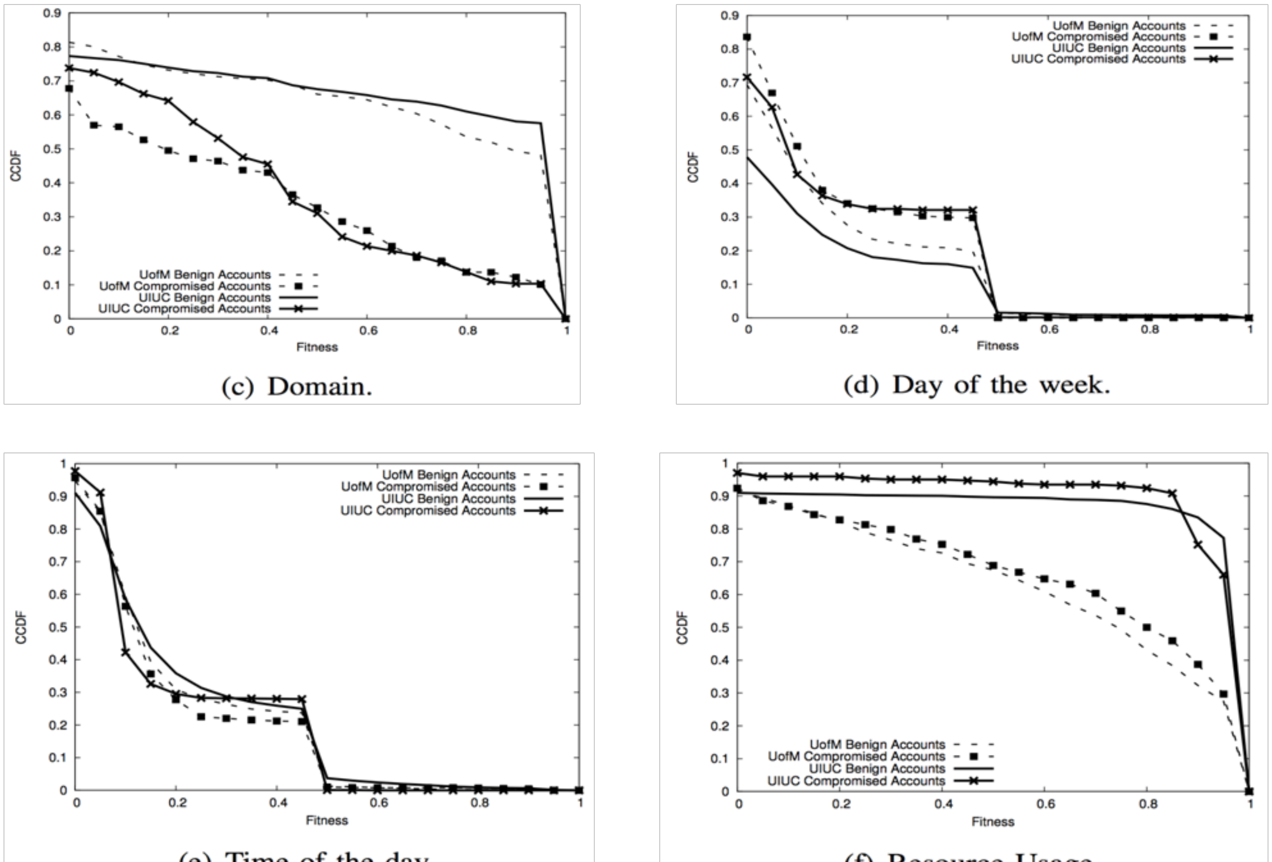


Institution	Website	% of benign	% of compromised
UofM	Library	5.83%	44.66%
	Web Mail	34.63%	28.72%
	Course Portal	5.93%	0.78%
	Remote Desktop	1.39%	0
	File Storage	1.22%	0



(a) UofM.

(b) UIUC.



(c) Domain.

(d) Day of the week.

(e) Time of the day.

(f) Resource Usage.

Figure 49: Complementary cumulative distribution function of profile fitness for benign and compromised accounts

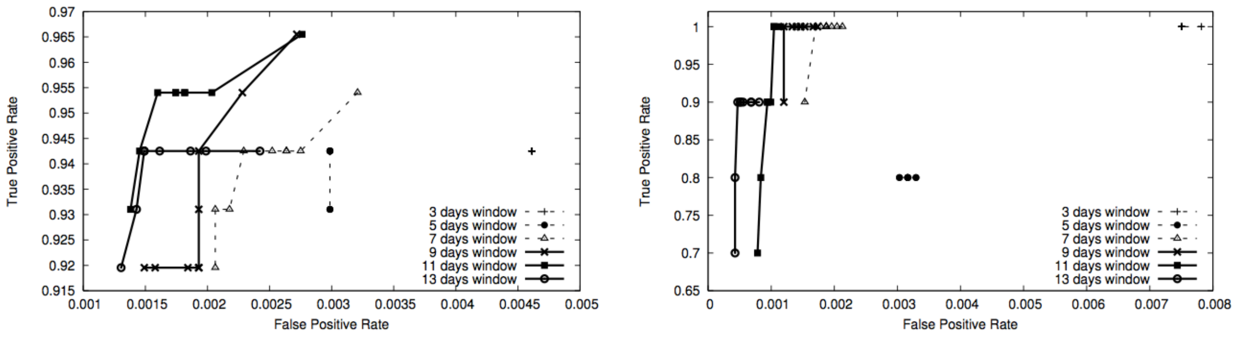
We then reviewed the performance of the resource usage-based features. As shown in Table 22, compromised accounts are more likely to use the VPN rather than web-based services for both universities. Again, the coefficients for the features of web-based service and VPN service at UofM are -0.38 and 0.38, respectively. However, the low proportion of web-based service usage at UIUC prevented us from computing meaningful performance results for the website usage.

Thus, Table 23 presents results only for UofM. Once again, the result matches our intuition that a significant proportion of illegitimate activity is linked to library access that has a coefficient of 3.53, while legitimate users accessed the web mail, course portal, and other academic websites more frequently.

Finally, Figure 49 shows the performance evaluation of profile-based features. The complementary cumulative distributions of profile *fitness* for the different features indicate that IP-related profile features (country code, ASN, and domain name) are effective in differentiating legitimate from illegitimate activity. However, timing and resource-related features provided poor performance, since the profiles collected for the benign and compromised groups of accounts are indistinguishable.

4.7.3.7 Model Evaluation

To evaluate the model, we conducted a cross-validation, which is the traditional technique for evaluating machine-learning algorithms. Given the limited number of positives, we use a fivefold cross-validation to ensure an adequate number of positive samples in each set. Since each user may have multiple feature vectors corresponding to multiple days, we grouped the feature vectors by user before partitioning the dataset. By doing so, we prevented artificially good results by removing situations in which different feature vectors were recorded in the training set and the evaluation set for the same user.



(a) UofM.

(b) UIUC.

Figure 50: ROC curves

An important parameter to choose is the length of the training window. In Figure 50, we show the detection results collected from five training window sizes. As mentioned before, our goal is to limit the average daily number of false alarms to two. Since UCAAS analyzes about 1,000 unique users per day, we need to achieve a false positive rate (FPR) less than or equal to 0.2%. Under that requirement, the best true positive rates (TPRs) that the system achieved was 95.4% at UofM with a training window of 11 days, and 100% at UIUC with a training window of 9 or 11 days.

4.7.3.8 Empirical Evaluation

We conducted the empirical evaluation by building a model from the training set that was then used to classify the validation set. At UofM, during two weeks starting from September 14, 2011, 126 unique users who never appeared in the training set were flagged. 124 of them were validated by the security team as compromised. The remaining 2 accounts had been shared with family members or friends living in foreign countries. Those results exceeded our expectations, since none of the flagged accounts were false positives. Also, there were no compromised accounts that were detected in other ways but not detected by UCAAS. Therefore, we can conclude that UCAAS achieves better detection recall than any other existing methods in the university.

At UIUC, we conducted the empirical experiment by running UCAAS on the validation dataset collected from July 9 to July 23, 2011. A total of 11 alerts were produced, reflecting 10 accounts already labeled for illegitimate activity in the training set. They appeared here again because those accounts were still compromised when the validation set was collected. The flagged account that was not part of the training set was validated as a true positive. These results are encouraging, because no false positive was generated and a new compromised account was discovered. We also checked with the security team to confirm that the system did not miss a compromised account reported during this period. However, because we collected the validation set at UIUC very close to the time we collected the training set, the lack of newly compromised accounts limits our conclusions regarding the overall accuracy of UCAAS at UIUC. The fact that UIUC has 10 times fewer active accounts than UofM likely plays an important role in explaining the large difference between the numbers of compromised accounts at the two universities. As mentioned in Section II, the difference in the number of active accounts is due to the difference in account expiration rules. Indeed, we found that half of the 124 compromised accounts detected at UofM were alumni or former employee accounts. In conclusion, results at both universities revealed excellent performance. We were surprised to see that at UofM, the model built in June offered perfect accuracy on September-October data, indicating that it remained effective even after a few months.

4.7.3.9 Lessons Learned

We learned that the temporal-spatial violation feature was the main reason for false positives generated during initial testing phases. The incorrect violations were due to three types of events: 1) users connected through more than one VPN client, 2) users accessing campus services via remote desktops, and 3) imprecise geolocation information.

The first two issues were mitigated by the fact that our approach combines complementary sets of features. A temporal-spatial violation has to be associated with additional suspicious behavior captured by other features to raise an alert. To address the last issue, of unreliable geolocation, we revised location-related features to work at the country level rather than the city level, and use a GeoIP database [39] that is accurate enough at the country level. However, we observed that attackers who own credentials from multiple institutions can login to one university account via the VPN of another university. By doing so, they can hide geographic location information and evade our detection method. Therefore, the city-level geolocation information is essential to covering those cases. As future work, we plan to integrate better geolocation lookup approaches [40] to increase the robustness of UCAAS.

4.7.3.10 Conclusions

Large academic institutions are exposed to the difficult challenge of protecting user accounts while supporting a wide set of services with limited security resources. This section presented the University Credential Abuse Auditing System (UCAAS), a machine-learning approach for automatic detection of account compromises that abuse the VPN service. It considers a large set of automatically generated features. These features are evaluated on their ability to identify illegitimate behavior. A logistic regression classifier is then used to flag accounts that are likely to be compromised. The system was trained and evaluated across two large universities and has been used by the operations team to identify a total of 125 compromised accounts in our two-week trial. Empirical validation shows that UCAAS offers high detection accuracy with no false positives across the two universities. This work is the result of an extensive collaboration, not only between researchers at two different institutions, but also between researchers and security analysts who deal with the issue of account compromise on a daily basis.

4.7.4 Architecture and Documentation of UCAAS

The algorithm presented in the previous section has been implemented in a production-ready prototype and delivered as a self-contained virtual machine. This virtual machine includes a log generator so that the system can be tested and understood prior to deployment. After launching the virtual machine, one can connect to port 8000 on the default interface with a Web browser to display the user interface. An SSH server also listens to port 22 and the default sudo-ready account has been setup:

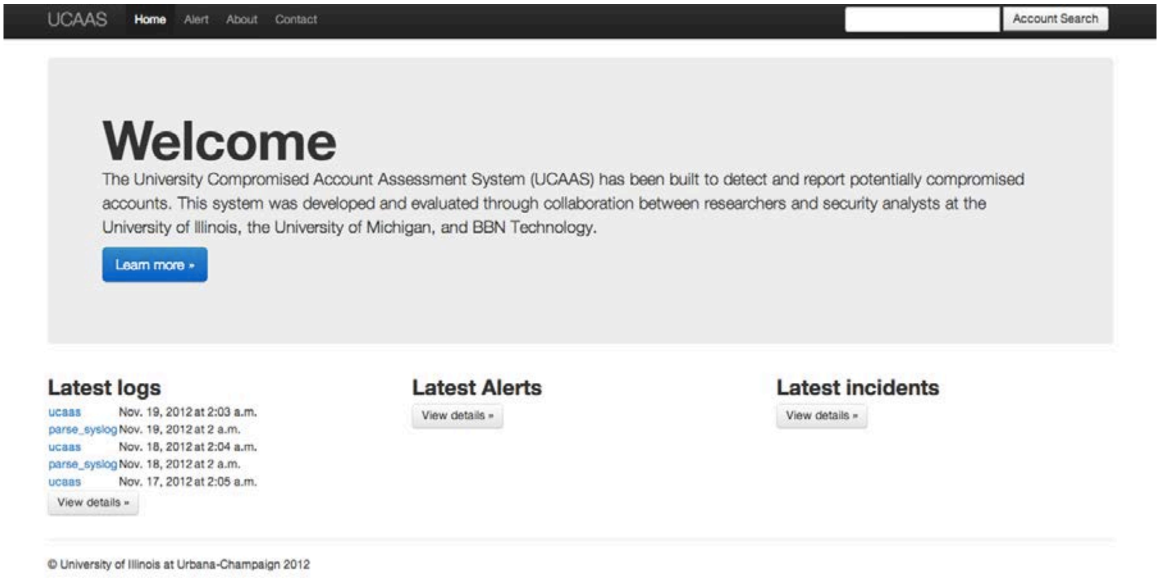
- Username: rgb
- Password: ucaas-demo

This section documents usage, architecture, installation, and maintenance of the system.

4.7.4.1 Usage

The system has been designed to run autonomously and a website is the only interface that operators should have to use. The welcome page is shown on Figure 51. The goal of this website is to allow operators to:

- Investigate authentication activity for a specific account through the **account** page
- Review alerts and incidents on a daily basis to correctly respond to compromised accounts.
    - **Alerts** represent feature violations and are described in the **alert** page
    - **Incidents** represent potentially compromised accounts and are detailed in the **incident** page
- Review script running logs to check that daily authentication activity processing scripts didn't generate warnings or errors through the **log** page



Welcome

The University Compromised Account Assessment System (UCAAS) has been built to detect and report potentially compromised accounts. This system was developed and evaluated through collaboration between researchers and security analysts at the University of Illinois, the University of Michigan, and BBN Technology.

Latest logs

```
ucaas        Nov. 19, 2012 at 2:03 a.m.
parse_syslog Nov. 19, 2012 at 2 a.m.
ucaas        Nov. 18, 2012 at 2:04 a.m.
parse_syslog Nov. 18, 2012 at 2 a.m.
ucaas        Nov. 17, 2012 at 2:05 a.m.
```

Latest Alerts

Latest incidents

© University of Illinois at Urbana-Champaign 2012

Figure 51: UCAAS interface and welcome page

4.7.4.2 Documentation: Architecture, Installation, Configuration, and Maintenance

1. **Goal**: The main design goal of UCAAS is to assist security teams by automatically flagging compromised accounts. The first requirement is that the system must be effective in detecting compromised accounts, even if the illegitimate activity is stealthy. UCAAS leverages a combination of features in which illegitimate activity is most likely to manifest.
2. **Overall Design**: UCAAS detects suspicious accounts based on authentication logs collected from university systems. Since VPN is used as an entry point by attackers, we first filter authentication activity to keep only logs generated by users who accessed the VPN at least once. The second step consists of extracting and analyzing the set of features for the daily activity of each account. The activity captured by the different features describes the behavioral, geographical, and topological pattern information, as well as possible deviations from historical profile data. Finally, a classifier runs on the feature vectors to determine whether an account is compromised or not. The model used by the classifier is trained and built automatically from the past n days of authentication logs. This training dataset is made of both known compromised and legitimate accounts, and is updated dynamically over time.

4.7.4.2.1 Architecture and Requirements

The software has been developed in Python and runs on Linux. It also contains a Java component to run a machine-learning algorithm provided by the Weka library. The application runs a Syslog server continuously and then analyzes authentication logs offline, on a daily cycle, and reports accounts for which suspicious activity has been detected. Figure 52 shows a high level overview of the architecture.

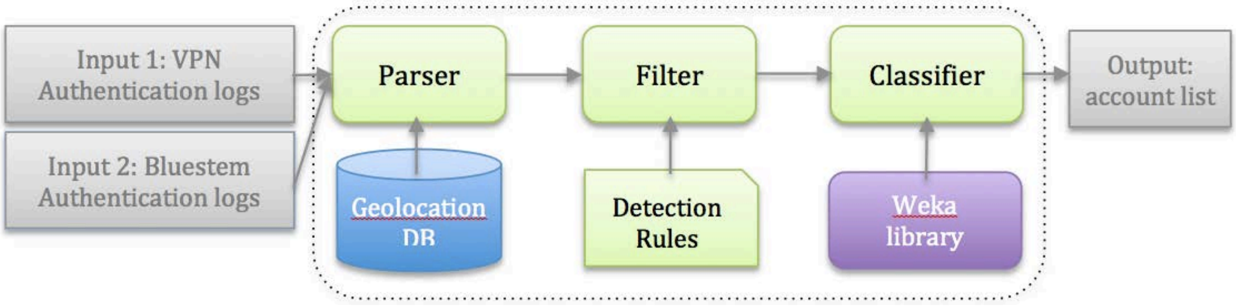


Figure 52: Architecture overview

The parser and filter modules have been developed in Python. The classifier module is in Java. The parser store logs to a SQL database. The geolocation database is downloaded offline from maxmind.com on a monthly basis. The configuration parameters include thresholds for the detection rules and for the classifier. VPN authentication logs can currently be parsed from Aventail, Cisco and Nortel devices, and should be filtered to keep only message logs having remote IP address information. Bluestem authentication logs should also be filtered to keep only login messages. The format of those logs is presented in the next section.

The diagram in Figure 53 (in the next page) describes the internal scripts and their dependencies.

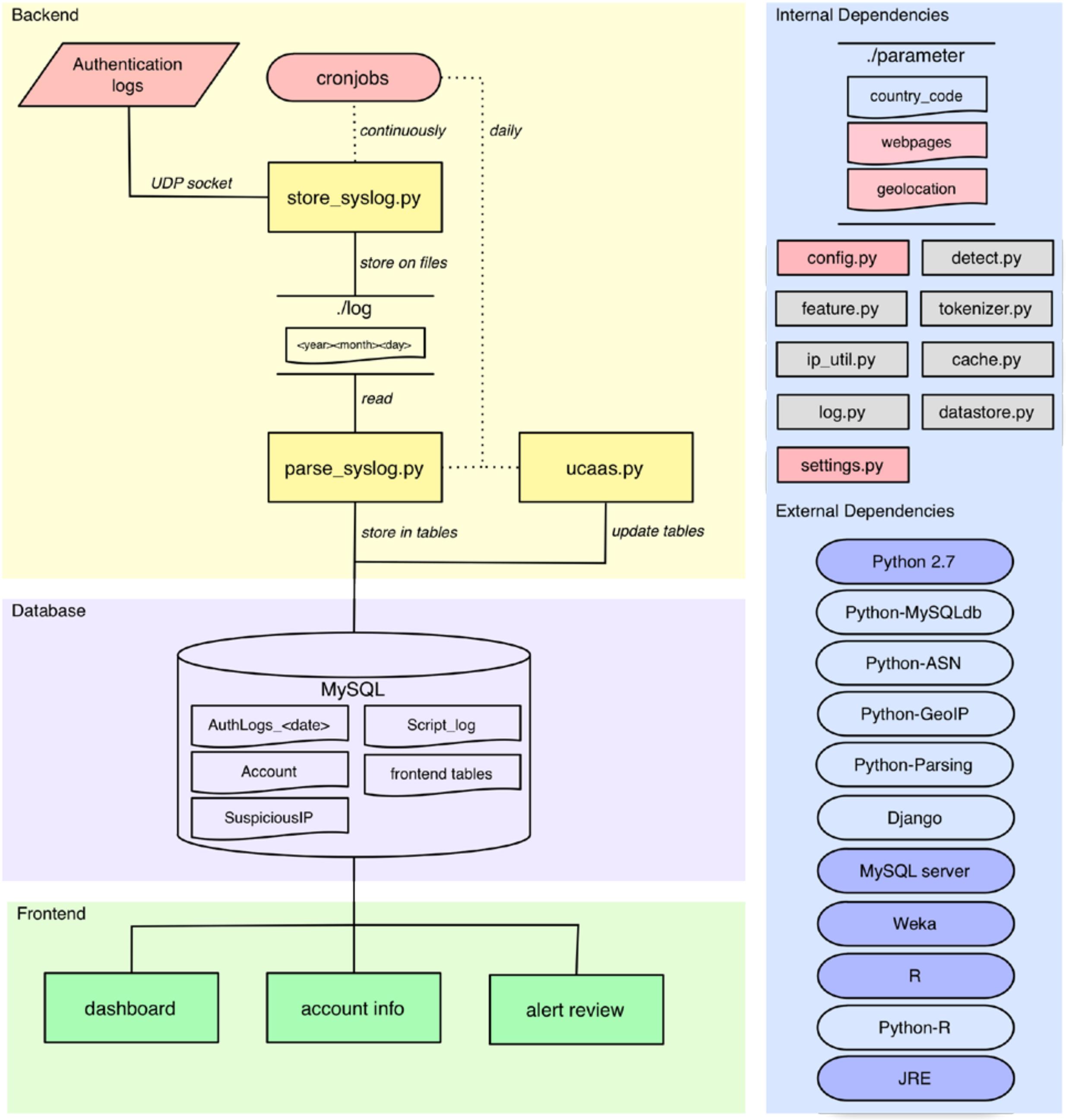


Figure 53: UCAAS internal architecture, components, and dependencies

4.7.4.2.2 Log Format

Parsers have been written for the following devices. New parsers can be easily added by editing the script `parse_syslog.py`. This script uses the pyparsing library to simplify the development of new parsers.

4.7.4.2.3 Nortel VPN logs:

- Filter: "Message" "Contains" "physical address: remote"
- Field selection:
  - DeviceType “nortelvpn”
  - UserName
  - MessageTime
  - DestinationAddress

4.7.4.2.4 Cisco VPN logs:

- Filter: "Message" "Contains" "Payload: Address"
- Field selection:
  - DeviceType “ciscovpn”
  - UserName
  - MessageTime
  - SourceAddress

4.7.4.2.5 Aventail VPN logs:

- Filter: "Action" "Equals" "logged in"
- Field selection:
  - DeviceType “aventail”□
  - UserName
  - MessageTime
  - SourceAddress

4.7.4.2.6 Bluestem web proxy logs:

- Filter: message selection auth > successful > login
- Field selection:
  - UserName
  - MessageTime
  - SourceHostname
  - Message

4.7.4.3 Detection Features

4.7.4.3.1 IP features

- CC: Country code
- ASN: Autonomous System Number
- Domain: Hostname and domain name

4.7.4.3.2 Resource features

- Service usage: Organization website access time, duration

4.7.4.3.3 Violation features

- IP violations: Same IP connecting to multiple accounts
- Time/Space violations: Same account being used in two different places
- Resource violations: Accounts using resources related to compromise

4.7.4.3.4 Profile features (Activity patterns learned over time)

- Country
- Dayoftheweek
- Domain_name
- Timeoftheday
- Usage
- ASN

4.7.4.4 Installation

Hardware requirements: one computer with a network interface and sufficient disk space to store authentication logs. Operating system requirement: Unix (e.g., Linux)

4.7.4.4.1 Installing Dependencies:

- Python 2.7: http://www.python.org/getit/releases/2.7/
- Python-MySQL 1.2.2: http://mysql-python.sourceforge.net/
- PyASN 1.2: http://code.google.com/p/pyasn/
- PyGeoIP 0.2.2: http://code.google.com/p/pygeoip/
- PyParsing 1.5.6: http://pyparsing.wikispaces.com/
- Django 1.4.1: https://www.djangoproject.com/
- Rpy 2.2.2: http://rpy.sourceforge.net/
- R 2.15.1: http://www.r-project.org/
- Weka 3: http://www.cs.waikato.ac.nz/ml/weka/
- Java VM 1.6: http://www.java.com/en/download/index.jsp
- Mysql server 5.5.27: http://dev.mysql.com/downloads/mysql/

4.7.4.4.2 Setting up the database:

After installing MySQL server, create a database “UCAAS” and a user “ucaas” having all privileges on the database “UCAAS.” The database, user name, user password, server IP/socket will be used in the configuration files of the backend and the frontend.

4.7.4.4.3 Editing configuration files:

The backend of UCAAS uses the single configuration file `config.py` where the following parameters should be defined:

Syslog parameters:

- syslog_port = (e.g., 514)
- syslog_host =
- syslog_storage = (e.g., "./log")
- data_storage = (e.g., "data")

MySQL parameters:

- db_host = "localhost"
- db_socket = "/tmp/mysql.sock"
- db_user = "ucaas"
- db_pass =
- db_database = "ucaas"
- table_prefix = 'AuthLogs'

Geolocation and Resource parameters:

- path_asn = './parameter/ipasn_20120703.dat'
- path_geoip = './parameter/GeoLiteCity.dat'
- tokenize = True (to anonymize user names)
- path_db = "./db/"
- path_cc = './parameter/country_code.txt'
- path_cc_lat_long = './parameter/country_code_latlon.csv'
- path_domain = './parameter/1st_level_domain.txt'
- weblist = './parameter/webpage_uiuc.txt'
- servicelist = List of services (e.g., 'Bluestem', 'VPN')
- websites_linked_to_incident = List of websites often targeted by attackers (e.g., 'library.illinois', 'library.uiuc')

The frontend uses the single file `frontend/frontend/settings.py` where the DATABASES section should be configured similarly to the backend.

4.7.4.4.4 Editing parameter files

The folder parameter contains the following files that should be completed:

- webpage_uiuc.txt: to be filled with websites internal to the organization in which the system is deployed and that may appear in the authentication logs
- GeoIP information: download from Maxmind the following files and uncompress them in ./parameter:
  - http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
  - http://www.maxmind.com/download/geoip/database/asnum/GeoIPASNum.dat.gz
  - http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz

4.7.4.4.5 Configuring cronjob

Add the following entries to your crontab:

```
* * * * * /run_syslog_server_watchdog.sh & * * * * *
/run_web_server_watchdog.sh 0 2 * * * /nightly_process.sh
```

The script `run_syslog_server_watchdog.sh` will open a UDP port (according the backend configuration defined in `config.py`) and wait for Syslog logs to store on disk. This script should have write access to the folder in which UCAAS is installed. The script `run_web_server_watchdog.sh` checks every minute if the MySQL and Python Web servers are running. If not, new processes are launched.

Finally, the script `nightly_process.sh` runs every night at 2 AM to parse the last day of authentication logs received, and then run the core engine (`ucaas.py`) to import everything in the database, to generate features, to re-train the machine learning model, and to run the classifier in order to generate incidents.

An additional script `run_mysql_watchdog.sh` can also be used but it requires root privileges so it should be added to the root crontab:

```
* * * * * /run_mysql_watchdog.sh
```

4.7.4.4.6 Verifying that UCAAS is working

The script `run_web_server_watchdog.sh` will ensure that the Django web server is running. One can then point a web browser to http://localhost:8000 to configure a user for the frontend and to open the dashboard in which script information and results of analysis will be displayed.

4.7.4.4.7 Folder organization

Three folders are being updated by the system within the UCAAS folder:

- **./log**: contains the raw syslog files received by the local Python syslog server and also the log files for parse_syslog.py and ucaas.py. Files are organized per year/month and named `data_[year]-[month]-[day].[log,info,error]`
- **./data**: contains the processed authentication activity, with parsed syslog files (.data extension), machine learning feature files (.arff extension), and user list file (.user extension), and machine learning classification file (.result extension). Files are organized per year/month and named `data_[year]-[month]-`

[day].[data,result,user,arff]. Each monthly subfolder also contains the latest machine learning training model (training_model.[arff,info,user,weka]).

- **./db**: contains cache files used to speed up geolocation, ASN, and hostname lookups. Also contains token/user ID maps if the tokenizer (anonymizer) is enabled in config.py.

4.7.4.4.8 Periodic maintenance tasks

The following tasks should be conducted on a monthly basis:

1. *Cleaning up log files*: by compressing or deleting log files in the ./data and ./log folders in order to prevent the disk space from being saturated. The cache files in the ./db folder can also be cleaned up when they become too large (beyond 10 MB, lookup operations will take too much time).
2. *Cleaning up MySQL*: the Account and SuspiciousIP tables in the UCAAS database will grow over time. Entries older than few months can be back up to files and deleted to prevent tables from being too large.
3. *Updating lookup files*: by running the geolocation and ASN database update script update_GeoIP_db.sh to refresh lookup files in the ./parameter folder.

4.7.5 Next steps: automated response system

The focus of this project has been to study the problem of account compromise in academic environments and to design a detection framework based on a machine-learning approach. The results from an empirical validation of this framework show high accuracy and the security teams at the University of Illinois and the University of Michigan are interested in deploying the framework in production.

While such a detection framework already assists security administrators in identifying malicious activity, it does not alleviate the task of manually responding to incidents. Appropriate and timely incident response is indeed a critical part to enable IT architecture to be resilient, but it can be so complex that it is often under the entire responsibility of a human operator. The issue is that humans are resource-limited or too slow to properly react to the growing number of cyber attacks. A solution is to implement automated response actions, but those are usually extremely simple and organizations are reluctant to deploy more complex actions. Therefore, we believe that a valuable next objective would be to understand the reason behind this gap, and to introduce a practical framework to bridge it.

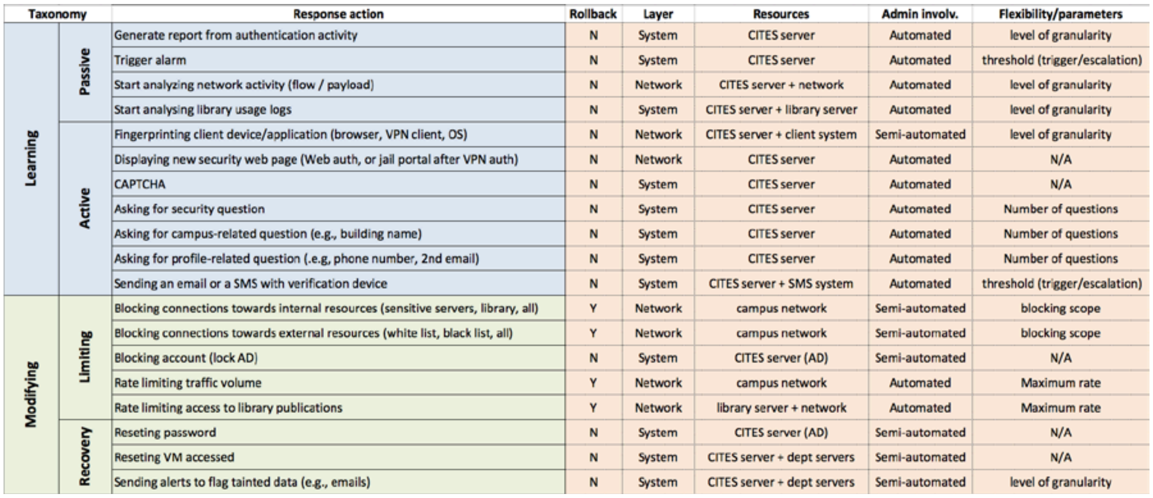
Key research questions under this objective would include:

- Where have automated response actions been successfully implemented?
- What are the characteristics of a successful/failed automated response action?
- What are the barriers to design and deploy automated response actions?
- Where are human operators indispensable?
- What are the research challenges to push automation forward and minimize human involvement in responding to cyber incidents?

- Can response actions be automatically generated based on a given system architecture and a given security policy?
- Can we build a tool to advise security administrator in deploying optimal response actions?

We suggest an approach in three phases to start answering those questions. The first, is to develop a taxonomy of possible response actions. The second is to implement a simulation environment to evaluate those actions under various conditions. The third, is to design a cost model that can be used at the core of the simulation, and then to be deployed within the decision-making engine for response and recovery.

Figure 54 describes a first attempt at listing all possible responses following the detection of a compromised account. The taxonomy uses a two-level hierarchy in which responses are classified first by their objective (learning more information about the incident, or modifying the system to recover to a secure state) and then by their mode of operation (active or passive in case of learning, and recovery or limitation in case of a modification). On the right-hand side of Figure 54, additional characteristics are provided for each response action, in order to understand their implication and the cost of their deployment.



Taxonomy		Response action	Rollback	Layer	Resources	Admin involv.	Flexibility/parameters
Learning	Passive	Generate report from authentication activity	N	System	CITES server	Automated	level of granularity
		Trigger alarm	N	System	CITES server	Automated	threshold (trigger/escalation)
		Start analyzing network activity (flow / payload)	N	Network	CITES server + network	Automated	level of granularity
		Start analysing library usage logs	N	System	CITES server + library server	Automated	level of granularity
	Active	Fingerprinting client device/application (browser, VPN client, OS)	N	Network	CITES server + client system	Semi-automated	level of granularity
		Displaying new security web page (Web auth, or jail portal after VPN auth)	N	Network	CITES server	Automated	N/A
		CAPTCHA	N	System	CITES server	Automated	N/A
		Asking for security question	N	System	CITES server	Automated	Number of questions
		Asking for campus-related question (e.g., building name)	N	System	CITES server	Automated	Number of questions
		Asking for profile-related question (.e.g, phone number, 2nd email)	N	System	CITES server	Automated	Number of questions
		Sending an email or a SMS with verification device	N	System	CITES server + SMS system	Automated	threshold (trigger/escalation)
Modifying	Limiting	Blocking connections towards internal resources (sensitive servers, library, all)	Y	Network	campus network	Semi-automated	blocking scope
		Blocking connections towards external resources (white list, black list, all)	Y	Network	campus network	Semi-automated	blocking scope
		Blocking account (lock AD)	N	System	CITES server (AD)	Semi-automated	N/A
		Rate limiting traffic volume	Y	Network	campus network	Automated	Maximum rate
		Rate limiting access to library publications	Y	Network	library server + network	Automated	Maximum rate
	Recovery	Reseting password	N	System	CITES server (AD)	Semi-automated	N/A
		Reseting VM accessed	N	System	CITES server + dept servers	Semi-automated	N/A
		Sending alerts to flag tainted data (e.g., emails)	N	System	CITES server + dept servers	Semi-automated	level of granularity

Figure 54: Taxonomy of response actions following the detection of account compromise

We then suggest leveraging a simulation environment to understand the impact of response action without having to actually instrument a production system. The objective is, given a situation and an environment, to be able to compare the impact of an array of possible actions, taken manually or automatically. Figure 55 presents a hybrid simulation model in which we leverage access to live data in order to provide highly realistic results. The goal is to use experimental data to feed the simulation with realistic parameters and incident scenarios. The top part of Figure 55 is essentially the UCAAS system running in production, while the bottom part is a simulation model than can be implemented in Möbius [42]. The role of the simulation is to compute the cost of all possible response actions given information collected about the incident, the state of the system, and a cost model (outlined in Figure 56). The optimal can then be suggested to an operator who can choose to implement it or to implement a different response. After using the system over several weeks, one can then compare the efficiency of the automated decision-making module by reviewing the costs of operator-selected actions versus model-selected actions. One can even uses data collected after the incidents to label the incidents as false or true positive, in order to adjust the cost computation.

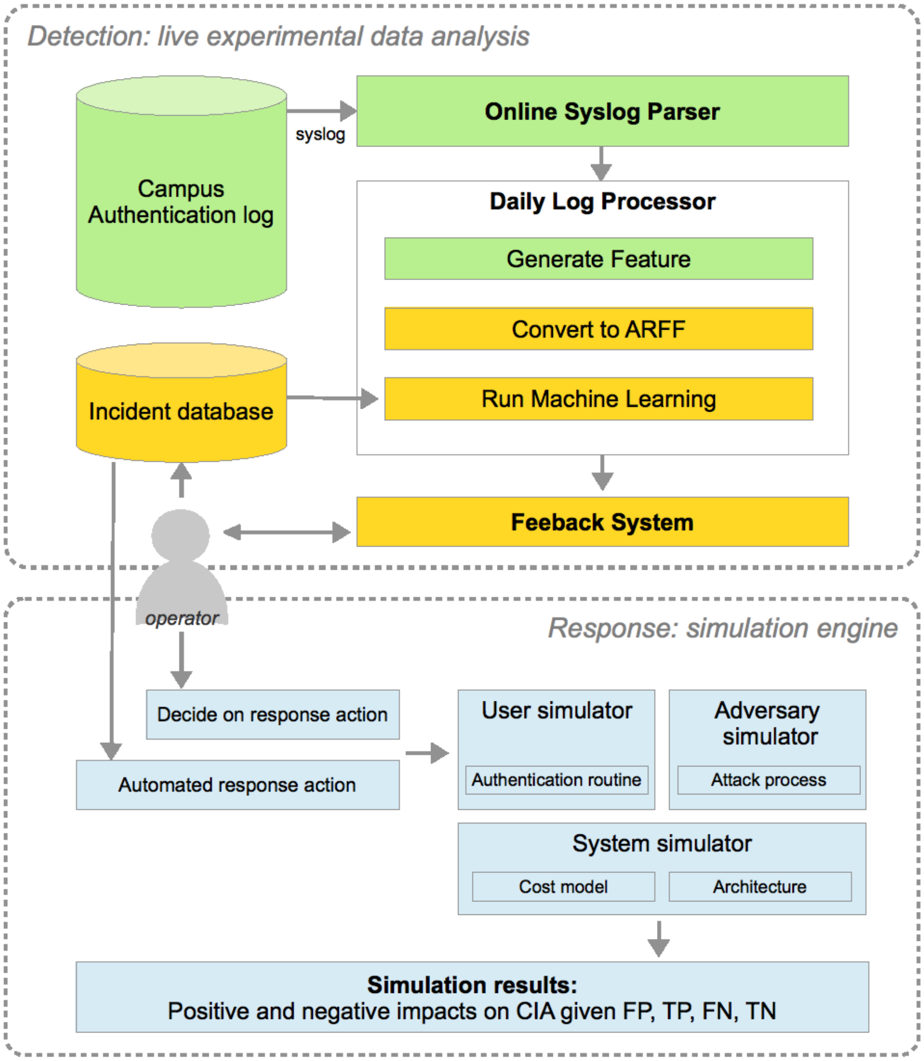
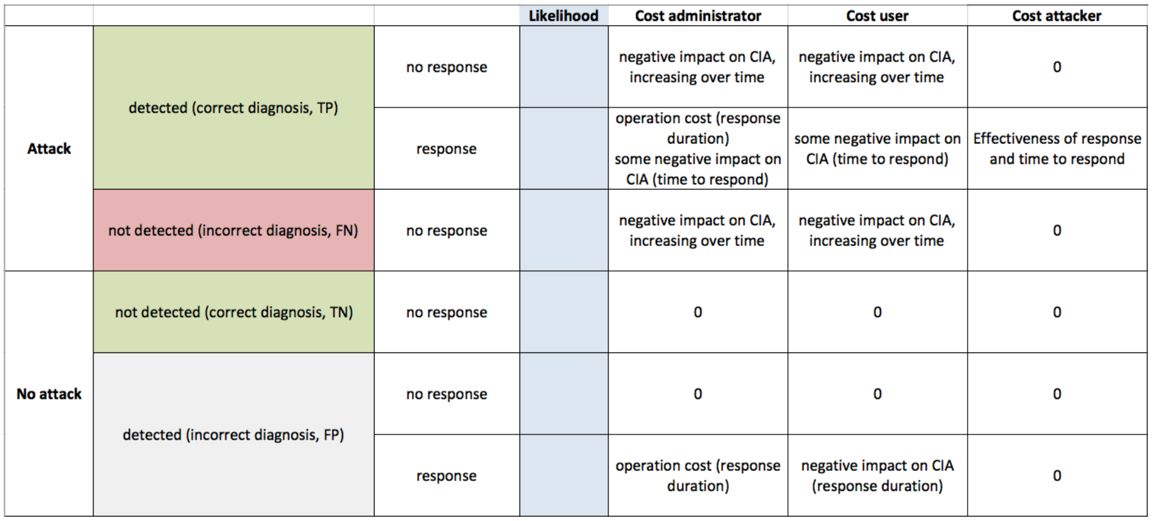


Figure 55: Combination of a simulation environment with UCAAS to understand the cost and benefits of response actions



			Likelihood	Cost administrator	Cost user	Cost attacker
Attack	detected (correct diagnosis, TP)	no response		negative impact on CIA, increasing over time	negative impact on CIA, increasing over time	0
		response		operation cost (response duration) some negative impact on CIA (time to respond)	some negative impact on CIA (time to respond)	Effectiveness of response and time to respond
	not detected (incorrect diagnosis, FN)	no response		negative impact on CIA, increasing over time	negative impact on CIA, increasing over time	0
No attack	not detected (correct diagnosis, TN)	no response		0	0	0
	detected (incorrect diagnosis, FP)	no response		0	0	0
		response		operation cost (response duration)	negative impact on CIA (response duration)	0

Figure 56: Definition of the various categories used for a response cost model

4.7.6 Concluding Remarks

We described the development of an anomaly-based assessment and auditing mechanism that can identify account compromise in large organizations with high accuracy. First, a rigorous approach to build and evaluate security metrics with respect to the issue of account compromise has been defined. This approach covers the data collection, definition of hypothesis, and selection of appropriate statistical tests to understand the key characteristic of the security issue under study. Second, we explored the design and efficiency of a variety of data mining features for which suspicious activity would likely manifest. The architecture of the auditing and assessment mechanism is designed to accommodate new set of features, new environments, or even new attack processes. Third, we demonstrated how the prototype can be deployed within an university environment. It only requires authentication logs and runs passively every night to output a list of accounts for which suspicious activity was recently detected. Finally, the accuracy of the system has been evaluated at two large academic institutions over several weeks and led to the successful detection of unknown compromised accounts with less than two false positive per day.

5.0 CONCLUSIONS

The recently concluded phase of the QIAAMU project is highly successful both in terms of theoretical advances and developing a working prototype system. A level of evaluation was also performed on the prototype, establishing the viability of the technical approach for runtime assessment and tradeoff developed in this project. Research results were disseminated by scientific publications, presentations at conferences and demonstrations as BBN and at AFRL. This project supported a number of interns, as well graduate students and post doctorate researchers over the years indoctrinating them with the issues and ideas underlying this research effort. At BBN, this project facilitated and enabled young and early-in-their-career researchers to use their theoretical skills in a new context and translating theoretical ideas into working prototypes. Therefore, this project is quite successful from the human resource development perspective as well.

One of the main technical contributions of this project is the methodology and supporting services and software engineering infrastructure for assessing the assurance state at runtime. This approach, unifying QoS and security, and instead of absolute quantification seeking to compare and contrast desired vs. delivered levels broke new ground in how security of distributed information systems and networks are evaluated. The other major contribution is an intelligent and mission-aware way to manage the tunable aspects of the system, driven by tradeoffs that span the QoS and security requirements of various stakeholders, and attempts to satisfy as many of these requirements as best as it can within the confines of the resources and security capabilities available at that moment. In the span of four years, the project covered quite a bit of ground starting from formalizing the problem space, solution approach, methodology for collecting information to prototyping supporting services, developing mission scenarios, integrating the key mechanisms within a distributed system and mission scenarios, to performing injection-based test and evaluation.

The results of the internal test and evaluation indicate that runtime assessment and management of QoS-security tradeoff is (a) beneficial and (b) viable. While promising, these results are from an internal evaluation based on a realistic but not an actual mission emulated in a virtual environment. Additional and more rigorous evaluation in the context of a real system supporting a real mission is clearly needed. One other point to be emphasized about these results is that the focus of the evaluation was more on the assessment and tradeoff management algorithms and mechanisms, assuming the required inputs are provided and/or available. These inputs include system information (cause and effect dependency), QoS and security requirements of the mission, etc. It will take a different kind of evaluation to test whether the assumption about the nature and availability of such information is valid or not. We expect our next round of evaluation and demonstrations will provide the opportunity to address these issues.

Despite the successes, a number of challenges still remain. Many of these are inherent in the nature of R&D: in order to investigate a problem or a solution approach, researchers often make simplifying assumptions that enable focusing on key sub-problems first, faster prototyping and incrementally establishing the feasibility of a complete solution. QIAAMU is no exception. The simplifying assumptions made by this phase of QIAAMU are acceptable for an R&D prototype, but not for a deployment ready technology. For example, one current assumption is that a QIAAMU agent (blackboard) will always be collocated with any sensor providing the raw measurement/observation or actuation control representing a tunable aspect of the system. This eliminates the need to worry about the security and trust issue involving the information exchange or command-control interactions between a blackboard and the sensor or actuation control mechanism. This assumption may not always be valid, and the QIAAMU interaction patterns must provide adequate security. Another assumption is that blackboards never get completely disconnected from their peers (other blackboards or local sensor or actuation control mechanisms)—their communication may degrade, but they never get isolated. In a real situation, especially in a hostile environment this assumption may not be valid, and QIAAMU agents must be able to provide continued service despite being isolated or partitioned. Now that the QIAAMU technology firmly established the viability of runtime mission-oriented assessment and tradeoff-driven adaptation, the time is right to investigate ways to address the limitations imposed by these simplifying assumptions.

Finally, although the QIAAMU technology is deployed in a representative system and the entire prototype is successfully delivered, deployed and demonstrated in an outside laboratory (AFRL) environment, QIAAMU is still an R&D prototype. Additional work in the area of usability and robustness of implementation is needed to mature the capability further. For example, QIAAMU offers a reasonable user interface for runtime monitoring of the QoS and security assessments and tradeoff actions using the QIAAMU GUI and through the AFRL ACS UDOP viewer. But use of QIAAMU before runtime is not equally streamlined. The process of configuring the blackboards is configuration file-driven and requires deep familiarity of the QIAAMU technology. Even though a lot of the code underlying the runtime blackboard instances are reused and automatically extracted or generated, the blackboards are specified by XML files in the current implementation. There are considerable opportunities to make the pre-runtime user experience i.e., the process of defining the blackboards better. This will involve better support and possibly graphical means of importing (i) mission stakeholder's QoS and security requirements and preferences, and (ii) system dependency in the form of cause and effect networks. We plan to explore next these issues next.

REFERENCES

[1] Schantz, R., Loyall, J., Rodrigues, C., Schmidt, D., Krishnamurthy, Y., and Pyarali, I., "Flexible and Adaptive QoS Control for Distributed Real-time and Embedded Middleware," *Proceedings of Middleware Conference,* Rio de Janeiro, Brazil, (Jun 2003).

[2] Pal, P., *QIAAMU Metrics: Information Assurance System Conditions and Quality Descriptors*, Raytheon BBN Technologies, Technical Report Delivered to AFRL, May 2009

[3] Farinelli, A., Rogers, A., Petcu, A, and Jennings, N. R., "Decentralized Co-ordination of Low-power Embedded Devices Using the Max-Sum Algorithm," *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-agent Systems*, pages 639-646, Estoril, Portugal, (2008).

[4] Petcu, A. and Faltings, B., "DPOP: A Scalable Method for Multi-agent Constraint Optimization," *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 266-271, Edinburgh, UK, (2005).

[5] Petcu, A. and Faltings, B., "A Distributed, Complete Method for Multi-agent Constraint Optimization," *Proceedings of the CP 2004, 5th International Workshop on Distributed Constraint Reasoning*, (September 2004).

[6] Petcu, A. and Faltings, B., “MB-DPOP: A New Memory-bounded Algorithm for Distributed Optimization,” *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1452-1457, Hyderabad, India, (2007).

[7] Farinelli, A., Rogers, A. and Jennings, N., “Bounded Approximate Decentralized Coordination Using the Max-Sum Agorithm,” *Proceedings of the IJCAI-09 Workshop on Distributed Constraint Reasoning*, pages 46-9, (July 2009).

[8] Zivan, R. and Peled, H., “Max/Min-Sum Distributed Constraint Optimization through Value Propagation on an Alternating DAG,” *Proceedings of the 11th International Conference on Autonomous Agents and Multi-agent Systems*, Valencia, Spain, (June 2012).

[9] Fave, F. M.D., Stranders, R., Rogers, A. and Jennings, N., “Bounded Decentralized Coordination Over Multiple Objectives,” *Proceedings of the 10th International Conference on Autonomous Agents and Multi-agent Systems*, pages 371-378, Taipei, Taiwan, May 2011

[10] Xiang, Y., Chen, J. and Havens, W.S., “Optimal Design in Collaborative Design Network,” *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-agent Systems,* pages 241-248, Utrecth University, The Netherlands, (2005).

[11] Bishop, C. M. **Pattern Recognition and Machine Learning**. Springer, 1st edition, 2006

[12] Fave, F.M.D., Rogers, A., Xu, A., Sukkarieh, S. and Jennings, N. R., “Deploying the Max-Sum Algorithm for Decentralized Coordination and Task Allocation of Unmanned Aerial Vehicles for Live Aerial Imagery Collection,” *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 469-476, (14-18 May 2012).

[13] Bahi, J.M., Contassot-Vivier, S., Couturier, R. and Vernier, F. A., “Decentralized Convergence Detection Algorithm for Asynchronous Parallel Iterative Algorithms,” *IEEE Transactions on Parallel and Distributed Systems*,**vol-16**, pages 4-13, 2005.

[14] L’eaut’e, T., Ottens, B. and Szymanek, R., “FRODO 2.0: An Open-source Framework for Distributed Constraint Optimization,” *Proceedings of the IJCAI'09 Distributed Constraint Reasoning Workshop*, pages 160-164, California, USA, (July 2009).

[15] University of Michigan, “IT security incidents,” http://www.safecomputing. umich.edu/main/incident_report.html, 2012. Accessed November 2012.

[16] Agresti, A. **Categorical Data Analysis**, Wiley Series in Probability and Statistics. Wiley-Interscience, 2nd ed. 2002.

[17] Caballero, J., Grier, C., Kreibich, C., and Paxson, V., “Measuring pay-per-install: The commoditization of malware dis- tribution,” *Proceedings of the 20th USENIX Security Symposium*, pages 187–202, (2011).

[18] Faraway, J. J., **Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models**, Chapman & Hall, 1st ed. 2005.

[19] Franklin, J., Paxson, V., Perrig, A. and Savage, S., “An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants,” *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 375–388, Alexandria, VA, (2007).

[20] Holz, T., Engelberth, M., and Freiling, F., “Learning More about the Underground Economy: A Case-study of Keyloggers and Drop Zones,” *Proceedings of the 14th European Conference on Research in Computer Security,* pages 1-18, Berlin, (2009).

[21] Jagatic, T., Johnson, N., Jakobsson, M., and Menczer, F., “Social Phishing,” *Communications of the ACM*, **Vol. 50**, No, 10, pages 94–100, October 2007.

[22] Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., and Savage, S., “Spamalytics: An Empirical Analysis of Spam Marketing Conversion,” *Proceed-*

ings of the ACM Conference on Computer and Communications Security, pages 3-14, Alexandria, VA, (Oct. 2008).

[23] Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L. F., and Hong, J., "Teaching Johnny not to Fall for Phish," *ACM Transactions on Internet Technology*, **Vol.** 10, No. 2, pages 1-31, May 2010.

[24] Lennon, E. B. "IT Security Metrics," *NIST Information Technology Laboratory Bulletin,* August 2003.

[25] Orme, J. G., and Combs-Orme, T., **Multiple Regression with Discrete Dependent Variables**. Oxford University Press, 2009.

[26] Paxson, V., "Strategies for Sound Internet Measurement," *IMC '04: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 263-271, New York, NY, (2004).

[27] Ritchie,F., "Secure Access to Confidential Microdata: Four Years of the Virtual Microdata Laboratory," *Economic and Labour Market Review,* **Vol.** *2*, No. 5, pages 29-34, May 2008.

[28] Rossow, C., Dietrich, C. J., Kreibich, C., Grier, C., Paxson, V., Pohlmann, N., Bos, H. and Van Steen, M., "On the Soundness of Silence: Assessments of Malware Execution Studies," *Proceedings of the 33rd IEEE Symposium on Security and Privacy,* San Francisco, (2012).

[29] United States Department of Homeland Security, **A Roadmap for Cybersecurity Research**. 2009.

[30] Sharma, A., Kalbarczyk, Z., Iyer, R. and Barlow, J., "Analysis of Credential Stealing Attacks in an Open Networked Environment," *Proceedings of the Fourth International Conference on Network and System Security*, pages 144-151, Washington, DC, (2010).

[31] Sheng, S., Holbrook, M., Kumaraguru, P., Cranor, L. F., and Downs, J., "Who Falls for Phish? A Demographic Analysis of Phishing Susceptibility and Effectiveness of Interventions," *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, pages 373-382, New York, NY, (2010).

[32] Sommer, R. and Paxson, V., "Outside the Closed world: On Using Machine Learning for Network Intrusion Detection," *Proceedings of the 31st IEEE Symposium on Security and Privacy,* pages 305–316, Oakland, CA, (2010).

[33] Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C. and Vigna, G., "Your Botnet Is My Botnet: Analysis of a Botnet Takeover," *Proceedings of the 16th ACM Conference on Computer and Communications Security,* pages 635-647, New York, NY, (2009).

[34] Winters, P. R., "Forecasting Sales by Exponentially Weighted Moving Averages," *Management* Science **Vol. 6**, No. 3, pages 324-342, 1960.

[35] Young, J. R., "Academic Publisher Steps up Efforts to Stop Piracy of Its Online Products," *The Chronicle of Higher Education*, June 26, 2011.

[36] Zhang, J., Berthier, R., Rhee, W., Bailey, M., Pal, P., Jahanian, F. and Sanders, W., "Safeguarding Academic Accounts and Resources with the University Credential Abuse Auditing System," *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, Boston, MA, (2012).

[37] Bolton, R. J., "Statistical Fraud Detection: A Review," *Statistical Science,* **vol. 17**, pages 235–249, 2002.

[38] University of Waikato, “Weka 3 - Data Mining with Open Source Machine Learning Software,” http://www.cs.waikato.ac.nz/ml/weka/. Accessed December 2012

[39] *MaxMind*. http://www.maxmind.com/app/ip-location, 2011, Accessed December 2012.

[40] Wang, Y., Burgener, D., Flores, M., Kuzmanovic, A. and Huang, C., “Towards Street-level Client-independent IP Geolocation,” *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, (Mar. 2011).

[41] Berthier, R. et al., “Learning from Early Attempts to Measure Information Security Performance,” *Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test* (CSET 2012), Bellevue, WA, (2012).

[42] https://www.perform.csl.illinois.edu/wiki/index.php/Mobius, Accessed December 2012.

[43] http://code.google.com/p/coreemu, Accessed December 2012.

[44] Chong, J., Pal, P., Atighetchi, M., Rubel, P., and Webber, F., “Survivability Architecture of a Mission Critical System: The DPASA Example,” *Proceedings of the 21st Annual Computer Security Applications Conference*, pages 495-504, Tuscon, AZ, (December, 2005).

[45] Pal, P., Webber, F., and Schantz, R., “The DPASA Survivable JBI – a High-Water Mark in Intrusion-Tolerant Systems,” *Proceedings of the EuroSys Workshop on Recent Advances in Intrusion-Tolerant Systems,* Lisbon, Portugal, (March, 2007).

APPENDIX A – DISCUSSION OF QOS AND SECURITY ATTRIBUTES

This section contains a general discussion on attributes used for assessing QoS and security. It also establishes how instances of metric classes from our metrics taxonomy can be used to determine the assessed value for these attributes using example systems and mission scenarios. The organization of the discussion follows the grouping described in Section 4.1.2, attributes that relate to both QoS and security is discussed first, followed by attributes that exclusively relate to either QoS or security.

A.1 Attributes Related to Both QoS and Security

Availability: This attribute indicates whether information system components, including components performing business as well as defense functions are up and running and available for use by the mission stakeholders. Note that this attribute does not imply anything about confidentiality, integrity or timeliness of the services of the functions provided by these components. If a component is running, or the function or service it provides is accepting or responding to external requests then that component is available. If a component is able to accept or serve requests from only a subset of those who want to use it, the availability is partial or degraded.

Consider an international-scale logistics, transportation and delivery scheduling system that allows users to coordinate the use of transportation resources (such as planes, trucks, personnel, etc.) across multiple countries and Areas of Responsibility (AOR). This example service depends on a database backend that replicates data about global resources at each AOR, but keeps AOR specific data locally. If the database with an AOR fails or becomes otherwise inaccessible within or outside the AOR, the global logistics system will be able to provide certain services, while failing to service requests that need the AOR specific local data. Someone who wants to plan a mission in the AOR whose database is down, is out of luck- to him availability is zero, whereas for a stakeholder who is planning a global mission may be able to find a circuitous route and also possibly a slower response. The lesser quality route or slower response is not indicative of availability however—they are reflected in other QoS attributes we describe later.

The availability attribute is evaluated using metrics associated with service presence measured or observed at points of service delivery or use and from the point of view of the service consuming stakeholder. End user stakeholders usually are interested in business functions, and for them the measurements are best taken at the access point –whether the service is accessible or visible, accepting requests or responding to requests. The system administrator stakeholders are usually interested in the health and upkeep of defense mechanisms, and for them the measurements would best be taken where the protection mechanisms operate.

Consider again the logistics example and assume that AORs are the 6 DoD Combat Commands (AFRICOM, CENTCOM, EUCOM, PACOM, NORTHCOM, SOUTHCOM). Let us also assume that the overall availability of the logistics system is solely dependent on the up or down status of the AOR databases. Let us consider that we are assessing availability of the logistics system from the point of view of a planner at CENTCOM. If all AOR databases are up—i.e., are visible from CENTCOM, the availability of the logistics system is 1. If the database in PACOM crashes, the binary metric indicating the status of the PACOM AOR database (from the point of view of CENTCOM) will go down from 1 to 0. This will degrade the availability of the logistics system (once again, from CENTCOM's POV) from 1 to .83 (or 5/6th because 1 of the 6 binary metrics indicating availability of individual AOR databases is now 0 instead of 1).

In general availability should be assessed by taking a weighted-average of the relevant metrics concerning all of the mission-relevant components irrespective of whether those components perform business or defense functions. For instance, in the above example- the presence or up/down status (e.g., SVC STAT) of the application services that processes planning requests and the link that connects the service consumer and the service providers can also be a factor in addition to the up/down status of the AOR databases. In addition to the up or down status, the load on the link connecting the service consumer and the service provider or the load of the service provider hosts (e.g., RES STAT) will also impact the availability assessment. Finally, if service usage is subject to mechanisms like access control and authentication—then DEF STAT metrics indicating the up/down status of the defense mechanisms also impacts the availability assessment. In many cases, SVC STAT and DEF STAT can be combined in a STAT metric. For example, the application service may offer two levels of access—one with certificate based authentication and the other without any login, but requiring a HMAC token in each message. It is possible to think of a metric that returns 1 if the service is operating in the certificate based authentication and 0 otherwise, and another metric that returns 1 if the service is offering open access with HMAC token and 0 otherwise. Note that each of these metrics in essence implies that the service is up (SVC STAT) and specific defense configuration (DEF STAT).

The weighting could be assigned based on user priority and operational context. For example DEF STAT of an IA-critical firewall service has a very large weight relative to a less-critical e-mail grey-listing service used to filter out some (but seemingly never all) spam. The SVC STAT metrics of the application or database services have a larger weight compared to the RES STAT metrics of the link and service hosts.

Agility[3]: This attribute reflects the system's ability to reconfigure/adapt/respond to and accommodate changes. The changes a system may need to respond can be caused by new deployment configuration, evolving user priorities, new requirements, attack induce failures in the environment as well as within the system. Some of the changes are analogous to "compile time" i.e., the goal is to field the enhanced version of the system for a new mission, as opposed to the "run time" changes, where the goal is to modify the system while the mission is in progress. Although there are differences in scale, "time to readiness" can be one indicator of the system's agility.

For example, an information system that becomes ready to provide the most critical services after a catastrophic failure or a disruptive overhaul sooner is more agile than the system that becomes ready later. One simple way this can be assessed is to determine how long does it take to have a certain pre-determined percentage (say 90%) of the services from the time it became totally "unavailable" (i.e., no stakeholder was receiving any service). In some cases, a weighted average of services being up can be used, indicating the relative importance of the services to the specific stakeholder for whom the assessment is performed. More sophisticated assessments may consider measurements indicating service degradation (as opposed to merely considering the presence or absence of the service as described above).

[3] We note here that an entire DARPA program (META II) is devoted to derive metrics of this nature—specifically META II asks for adaptability metrics, that will indicate how easy or hard it is for system to adapt to new requirements, and changing situations both internal as well as external.

Consider again the international-scale logistics information system described earlier, but this time let us focus on the system owner stakeholder. If PACOM database becomes unavailable, only the planners that absolutely need PACOM data will suffer—other planners that do not require PACOM data can continue to plan. To address this concern, each AOR has a failover mechanism—if a AOR database becomes unavailable, the system switches to the standby database, but the switchover takes about 300ms during which no database is available on that AOR. Now also consider that the standby database can be out of synch with the primary for at most 10 seconds, that is, in the worst case, the standby database may lose transactions performed in the last 10 seconds. So, for events like a single AOR database failure, the system's agility can be assessed in terms of the two "metrics" mentioned above—failover latency and maximum skew. A simple assessment function that provides an upper bound can be 10.3s, a more realistic assessment can be (10000*.01 + 300) = 400ms assuming that the (historical or experimentally observed) probability of maximum skew is 1/100.

Agility is not limited to runtime adaptation only. The cost of upgrading the logistics system with new software versions, the down time caused by scheduled and unscheduled maintenance can be used to estimate how "agile" the system is for such scenarios (and for stakeholders who are responsible for those scenarios).

As explained above, the metrics used to assess agility can be measured or observed at different parts of the system (depending on the spatial scope the stakeholder is interested in) and at different times, possible collected over a period time. For business services, agility assessment would include measurements taken at user access points where users would observe how well services are being delivered during changes in operational context. For protection-services this assessment would include measurements taken where the protection mechanism operate – this could be at the server for some protection services such as access control, or this could be at the user access point for some protection services such as the distribution of antivirus updates.

There is some relationship between the agility and availability. Availability assessment is biased towards steady-state operational analysis of service delivery. On the other hand, agility assessment is biased towards the availability of the service during changes in systems' configuration. For many cases, agility can be assessed by taking a weighted-average of the downtime of services being delivered during reconfiguration where the weighting is assigned based on user priority and operational context.

A.2 QoS Attributes

Our prior work in middleware support for QoS managed adaptive behavior has identified a number of attributes for describing and managing QoS in distributed systems. These attributes provide a starting point for our present context.

Timeliness: This attribute reflects the ability of the information system to respond to service requests in a timely and/or regular manner. Among the measurements/observations (metrics) that can be used in the assessment of timeliness are:

- Whether processes can complete jobs, transactions in the amount of time scheduled or expected, or bounds on the time to complete jobs or transactions
- Deviation or jitter in the time to complete jobs or processes, especially for repeated jobs (such as streaming video services), where *regularity* is important.
- The round trip time for a service request

Consider again the distributed logistic information system example. When a user in a given AOR (NORTHCOM, for example) wishes to schedule a transfer of material to another AOR (CENTCOM, for example), the information system would have to query the backend databases for both AORs to see if there are any scheduled combinations of flights that can route the material from its origin in NORTHCOM to its destination in CENTCOM. The timeliness of such schedule operations depend on the time taken by the databases to process the submitted queries—if the query processing time is not bounded, i.e., increases as more users submit queries, then the timeliness of the schedule operation cannot be guaranteed. Similarly, the quality of the communication link from the planner to the AOR and the link between the AORs may also impact the timeliness of the scheduling operation: if the quality of the link is bad, packets may be dropped, causing retransmission.

Timeliness exhibits a property of "diminishing returns" for the end user. Having a "less than a minute" bound for the scheduling operation is better for timeliness than a "ten minutes" bound. A "ten seconds" bound is even better. However, the user-perceived increase in timeliness resulting from a decrease from an hour to a minute is generally much greater than the return of going from a minute to a second. Instead of using additional resources to decrease response times to be less than a minute, the information system may be better served by dedicating those resources to serving other queries. This realization offers the insight that in terms of tradeoffs, sacrificing timeliness (e.g., coming down from 10s to less than a minute) may be preferable if the saved computing or networking resources can be used to boost other attributes. To facilitate that, the assessment function must also reflect the "diminishing return". Consequently, we advocate assessing timeliness using a weighted average of sigmoid functions. (A sigmoid function f(x) =1/(1+x) exhibits the described property of diminished returns.)

Suppose the logistics information system in our running example should be able to respond to route plan requests within 1 minute. Suppose this distributed system can, on average, provide route planning in 0.8 minutes for all of the AORs except for the PACOM region where route plans are provided within 10 minutes due (possibly) to an increased operation tempo resulting from a crisis on the Korean peninsula. Also, because of the crisis the PACOM AOR is assigned a relative importance weight of 4 compared to the other AORs which have importance weights of 1. Using the assessment outlined above for this attribute in this simple scenario, the logistic information system would provide timeliness assessed at 4*1* (1/11) + 1*5*(1/1.8) = 3.14 (there are 6 AORs, one of them has weight 4, and its response time is 10 minutes, the remaining 5 has weight 1, and their response time is .8 minute).

Timeliness evaluation is highly dependent on stakeholder preferences. For example, if stakeholder service requests are not one way (i.e., when a return value, confirmation, etc., is expected) then averaging the Round Trip Time (RTT) is the appropriate approach to assessing this attribute. Conversely, if the service request is one way, then even though a confirmation sent to the requester may not be relevant, the time from request submission to job completion is the relevant measurement. These considerations can cause complications in practice because timeliness may have to be measured (or estimated) based on observations over multiple nodes in an information system including user access points and servers. In addition to the issue of distributed clock synchronization, measured values would need to be transmitted to where the assessment is performed, incurring further delay in assessment, and there may be additional security issues of maintaining the confidentiality of measured values in transit.

Fidelity: This attribute (often referred to as Accuracy as well) indicates the "quality" of results returned during service delivery. When data is transmitted to the user, part of the information may be lost, or the data reaching the consumer may become transformed due to compression and/or filtering. For example, data compression and filtering techniques used to reduce bandwidth usage could irrecoverably degrade the quality of the data transmitted between the data producer and data consumer. Additionally, the wrong or partial result may be returned.

For an example of fidelity assessment in the logistics information system outlined above, a helicopter pilot may request satellite imagery of an unimproved landing zone to aid his delivery of material. When the pilot requests this imagery (e.g., 1 mile radius around the landing zone lat/long coordinate), if the responses are compressed and low resolution black and white images (possibly to minimize the bytes transmitted to get some information as fast as possible over a wireless tactical communication link) the fidelity score will be lower than the case when responses contain color and high resolution images in that region.

Like timeliness, there is generally a decreasing return in fidelity from the end user point of view. So similar to timeliness, a weighted average of function like f(x) = x/(1+x), where x measures the accuracy of the information content, can be used to assess the fidelity score. The function f(X) = x/(1+x) is chosen because it reduces to 0 when x=0 and increases to 1 as x gets very large. In our continuing example, if imagery is provided to pilots with very high accuracy (say with a relative value of 1) for all AORs except for the PACOM AOR where accuracy has been degraded to 0.5 (PACOM is now sending low resolution, black and white images to save bandwidth and computing power, and the the 1 and .5 accuracy values for hi-res-color and low-res-b&w images are predefined), then using the AOR weightings described earlier, the fidelity assessment of the imagery request would be 4*1* (.5/1.5)+1*5*(1/2) = 4/3 + 5/2 = 3.83.

Similar to timeliness the evaluation of fidelity is highly dependent on stakeholder preferences, under certain mission conditions a low fidelity but more timely response may be preferred over a slower but high fidelity response. The metrics needed for evaluating fidelity may be collected at both servers where information is transmitted and at user access points where information is received.

Suitability: This attribute (also referred to as Precision in QoS literature) assesses how well the results returned by an information system satisfy requests from the user. When a user makes information requests, the information service might not have the correct information available (or easily available), so the information service could transmit substitute data that *nearly* satisfies the user request.

For example, in the logistics information system outlined above, a pilot may request a weather report for a certain latitude/longitude at a certain time. The information system might not have a weather report for the exact right time/place, but it could furnish the user a weather report for a nearby city (or several nearby cities) at the requested time. (In the civilian domain, consider airline reservation systems- most of the time a traveler will not get the exact schedule, departure/arrival city or price they want, but they get flights that are around the time specified, airports in the vicinity and a range of prices.) In this example, the assessment of the response would depend on how close the weather in the city is to the requested latitude/longitude. If the pilot requests the weather report for Fallujah, but is instead given a weather report for Ramadi, this generally will not be problematic due to the proximity and similarity in weather between these two Iraqi cities. Conversely, if the pilot requests a weather report for a point on a mountain range, but is instead returned weather reports for two nearby cities from the opposite sides of the mountain, this will be much less suitable due to the altitude differences between the request and the returned despite their relative proximity.

The example above suggests that a lot of auxiliary context information is needed to evaluate suitability (such as the coordinates of the location of the weather report, the altitude of the requested and reported location). Given such information (i.e., metrics), the assessment, which by necessity would also be context-specific because of the semantics involved, can be a "closeness of fit". In the above example, the underlying weather specific rules may be that if desired location is within 10 mile radius of the reported location and at the same altitude then the weather of the reported location is acceptable; and unsuitable otherwise leading to a binary value. This binary assessment function may be useful for someone who is trying to land or hover over the mountain top, but for someone who is planning a road trip to the requested location it may be too conservative.

Note that there may be "diminished returns" resulting from delivering information that is better than "good enough" as with several other attributes. As outlined in the example above, Fallujah and Ramadi have similar enough weather that it isn't worth the effort to store and maintain weather reports for both municipalities in the information system.

A.3 Security Attributes

Confidentiality: This attribute focuses on the assurance that information is accessible only to authorized entities. In other words, a high score on this attribute indicates that we are more assured that information will not leak out or is being exfiltrated. What information needs to be assured for confidentiality or which parts of the system is threatened by exfiltration attacks obviously will vary from one stakeholder to another, and from time to time within missions. Confidentiality is mostly about information, and can be threatened when the data is at rest, in motion or even when it is being processed. Encryption is the primary defense against such threats, which works for data at rest (encrypted storage) and data in motion (encrypted messages), but applications need to process unencrypted data (newer efforts are looking at computing with encrypted data—e.g., homomorphic encryption—but the technology is not ready for primetime yet). Access control and levels of isolation also contribute to assuring confidentiality. Specifically, for the case of data being processed, the level of isolation of the computing processes is an important consideration.

In the logistics information system example, the distributed AOR-specific database backend could be protected by placing their resident servers behind strong firewalls, encrypting all information on disk and encrypting all information communicated by the server. The metrics involved with the assessment of confidentiality of the database service therefore would include measurements on the strictness of the firewall, whether or how strong the level of encryption is for the communication between the database service and the consumers of the database service, and also the level of encryption on the disk. Clearly, these metrics are best measured at the points where the actions take place (e.g., communication, processing or storing and retrieval). The DEF STAT metrics described above can be used to enumerate the various possible "confidentiality protection" configurations the database service-consumer interaction can have. This gives us a basis for estimating whether the system is delivering the required/expected level of confidentiality assurance.

Integrity: This attribute provides an indication of whether the information conveyed has been tampered with in transit or whether a service offering or processing the information is corrupt. In the logistic information system example, when a pilot requests weather reports, whether those weather reports were corrupted during transmission, or whether the service provided incorrect weather report (determined by cross-checking other sources or other pieces of information) would affect the integrity of the pilot's mission role. Whether the information is corrupted in transmission can simply be determined by checking a signature. Whether the service provided incorrect information can be determined by semantic consistency checks (e.g., weather at this location cannot be like this at time, or cannot change this way in this short amount of time) or voting of answers from multiple sources.

Integrity assessment of information as well as the source of the information depends on measurements taken at places where information is consumed (end consumer, or interim stops in the path). Integrity of a service can be measured statistically based on data integrity maintained across multiple tasks.

Even though checking signatures or voting does not necessarily require semantic understanding, breach of integrity can lead to semantic inconsistency as well (e.g., an information source providing the lat/long that is in the Indian Ocean in response to a query that is supposed to return a CONUS location). In such cases, the sensors that measure or observe the information needed to assess integrity also need to be aware of the semantics. Therefore, like suitability, assessment of integrity can, under certain circumstances, also depend on a semantic understanding of the information being sought, provided or processed.

Authentication/Access Control Level: This attribute refers to the strength or strictness of the access control and authentication regime employed in the information system. An information system that is open to public networks (such as the Internet) should have a very different assurance assessment than information systems only exposed to protected intranets. Similarly, a web site that does not ask for users to log in will have a lower value for this attribute than those that do. Network configuration such as the existence, placement (range) and configuration of wireless access points contribute to the assessed value of this attribute. For instance, consider a service S1 that was originally available on a wired Intranet. Now consider the same network with a wireless access point added to it. The idea is to enable users moving around the offices to connect to the Intranet resources such as the service S1. If the wireless access point is configured to be open (or configured with a weak security) any laptop with a wireless card within range can now connect to the Internet, and try to use S1 whereas previously, one needed physical access to the network. The authentication/access control level for the configuration with the wireless access point should lower in this case than the original configuration. In the logistics information system example, the AOR nodes are in SIPRNET. Users need to authenticate using their CAC cards or must have an application (could be a browser) that has a DoD approved identity certificate. The CAC provides stronger assurance.

Assessment of this attribute depends on measurements and observations about the presence and strength of access control and authentication mechanisms. Like availability, authentication/access control metrics are generally measured on a services basis and not on the basis of individual service requests or on statistical measures over requests.

APPENDIX B – COMMONLY AVAILABLE METRICS AND THEIR USE IN ASSESSMENT

One of the objectives of this section is to motivate and establish that the metrics can be obtained from the system, and they can be used for meaningful assessment. Toward that end, we note that measurements and observations listed above are generally about one of the following:

a) Runtime aspects of the information system (such as service presence, throughput, etc…),
b) Inherent information system properties (such as the process and organizational maturity, etc…) or
c) Operational context that is not organic to the system (such as load, adversary capability, etc…).

Each of the above groups has different constraints/technical challenges for taking measurements and observations. We use this grouping to illustrate those issues and present the rationale why such measurements and observations contribute to runtime mission-oriented assessment.

B.1 Measuring Runtime Aspects of the System

The following measurements and observations are about what is running in the system, and their various characteristics, including those that change dynamically. They belong to various metric classes in our taxonomy as indicated in table 1.

Service Presence/Protection Presence: Metrics of this nature measure if services are active and able to serve requests. This metric captures whether business functions (i.e., services) and defensive functions (i.e., protection) are up and executing in the desired configuration. In a strict interpretation, a service is down when any of its individual functional offerings (for example, a service S may offer multiple methods m1,m2, m3…, and if any of the m1,m2,… are not available, S is considered down). Under a less strict interpretation, the service is considered up, as long as any of its offerings is available (i.e., S is down when all m1, m2, m3… are not available). Which interpretation to use in a context should be guided by whether the partial unavailability of service offerings provides any value to the stakeholder: a down service should have *no* value during down time.

Specific examples of metrics from this class include:

- Presence of an Authentication Service: Binary indicator of whether the service is running and/or is responding. Can be measured by a) looking for the corresponding process, b) looking for the open port associated with the service on the service host, c) by attempting to connect to the service
- Presence of a web server: Same as above, instead of “defense” this server actually participates in the “business” part of the system.
- Presence or level of Encryption: Indication of whether messages between a source and destination are encrypted, and/or what algorithm or key length is being used. Whether messages are encrypted or not can be observed by looking at the configuration settings of the service, or by using a probe. Which algorithm and key length is being used are configuration data, which in some cases can be queried from the service (containers), or obtained from configuration or the service itself by instrumenting the service code.
- Presence of connectivity: Can a service consumer (stakeholder) connect to/reach the services enclave or host? This is a binary indicator or (logical) link availability. It can be ob-

tained by a probe that attempts to make the connection (more accurate), or by monitoring the traffic (may be stale- past connectivity does not imply future connectivity).

- Presence of Secure Connection: Is SSL used, and which version? Can a client negotiate? This again can be obtained by instrumenting the server side code—that looks up configurations or proactively monitors the server, and provides the measurements and observations.

This metric is primarily affiliated with the availability and reconfigurability attributes. Availability (presence) of defense services impacts the level of assurance -- so this metric influences the remaining IA attributes as well.

These metrics should be assessed at points of service delivery as these metrics measure the presence or absence of service delivery. For business services, this measurement would best be taken at the user access point where users would observe how well services are being delivered. For defense services this measurement would best be taken where the protection mechanism operate – this could be at the server for some protection services such as access control, or this could be at the user access point for some protection services such as the distribution of antivirus updates. For most applications a user could aggregate the individual measurements and observations by using a weighted-average (e.g., a weighted average of the uptime of services hosted in a system) where the weighting is assigned based on user priority and operational context.

Response Time of services: Response time contributes to assessment of timeliness attribute associated with services. Timeliness for the system administrator stakeholder can be expressed as a "statistical" measure of the response time of the services hosted in the system operator's domain of control, especially if applications aren't bound by hard response deadlines. Response time and its relationship with timeliness contribute to availability attribute. A long response time beyond a threshold (e.g., timeout) can be equivalent to unavailability of the requested service. Below are some specific examples of Response Time metrics:

- Round trip time for obtaining the response or acknowledgement of completion of the last request for a specific user. This metric would be relevant where a user is most interested in a guaranteed upper time limit to return results.
- The average time taken by a server (such as a web server) to complete submitted requests/tasks for all requests.
- Same as above, but for a specific user or a specific service request.

When response time is measured in terms of job or task completion time, the measurement should take place near the service. When response time is measured from the service consumer's point of view (e.g., RTT) the measurement should take place at the point of service delivery.

Throughput: Throughput measurement is related to response time measurement: where the latter is measured in time units (e.g., average time to complete), the former is expressed as a rate (e.g., the rate at which regularly requested jobs are processed).

Throughput metrics are used to assess timeliness and availability. Examples of metrics in this class include:

- The rate at which data is transmitted over a TCP connection.
- The rate at which web pages can be served by a webserver.

Metrics in this class are best measured at service delivery points as these metrics are associated with user observations of performance as opposed to internal information system operations.

Accuracy of information and services: Accuracy metrics measure or observe aspects of the information being exchanged or how the services are performing. It can be a boolean indicating whether the right answer was returned/service functioning correctly, or a real number indicating how close the returned response is to the "optimum" response, whether additional, unneeded information is returned.

This metric is mostly used to assess the fidelity and suitability attributes, but "inaccuracy" can also imply loss of integrity or malfunction. If the integrity of a defense mechanism is impacted, it also indirectly impacts the purported purpose of the defense mechanism. For example, a system may have a business function replicated where a replication management subsystem is responsible for managing the lifecycle of the replicas. In this setting, the replication management mechanism contributes to the availability of the business function. Inaccuracy or malfunctioning of the replica management system will impact the availability of the business function provided by the replicas.

Specific accuracy metrics include:

- Binary value indicating compliance or preservation of invariants by a service or by the information content, such as
    - The response provides different types and number of information than what is expected. In most cases this can be observed by having access to and watching for the appropriate exception.
    - Result from a computation did not return the level of precision expected (3 digits after decimal instead of 6).
    - The response provided in a format that the requester cannot process (impacting suitability attribute)
- Real value indicating a closeness of fit/distance between desired and delivered, where desired is a property that is pre-specified, and delivered can be computed from the information conveyed.
    - The map returned is not exactly covering a 10 mile radius of given latitude/ longitude.
    - The "sparseness" (what sparseness means is context sensitive) in the information provided—e.g., in response to a request for historical records between two dates, the server provides data for a fraction of the years in that range (impacting the suitability attribute).
- The retrospective false positive rate of a defense mechanism, typically a detection mechanism such as an IDS or a DDOS detection mechanism. Measuring this false positive rate assumes that there is some way to detect when the mechanism mis-categorized, which can often be done only manually and require observations and measurements made at multiple parts of the system. A high false positive rate implicates the integrity of the defense mechanism itself, and indirectly impacts the availability of the service or system (spatial scope) that it was defending/monitoring.
- The percentage of responses which are relevant to a user by a service, such as a Semantic Web triple-store responding to SPARQL queries. Metrics in this class are feasible only when the responses returned by an information system can be evaluated by some means.

Even if the responses can be evaluated, this evaluation may be computationally expensive, and this should be performed selectively or even retrospectively to avoid overloading the information system. Measuring metrics in this class will generally include observations at both servers where information is transmitted and at user access points where information is received.

Smoothness of service delivery: Smoothness metrics measure the variance or regularity of service delivery. In that sense this is a derivative of timeliness measurements. Smoothness measurements directly influence assessment of fidelity (e.g., in voice or video applications). Smoothness measurements are often included in assessment of other attributes such as timeliness and integrity, for example, by including a restriction on variance in the response time in the timeliness requirement, or arguing that large variance in a service's response time is an indication of the services lack of integrity. We argue that this view is incorrect—despite large variance the response time may be within acceptable range and the request/response may not contain any corruption, and the requirements on the variance are better treated as fidelity requirements.

The primary examples of smoothness measurements are derived from the variation in response time. Examples include:

- The standard deviation in the video packet delivery time.
- Packet drop rate
- Variance in completion times of scheduled tasks

Metrics in this class are best measured at the places where timeliness measurements are taken.

Resource Allocation: Resource allocation metrics measure the "goodness" of resource allocations. It is a symptom of how efficient resources are being used to maintain a sufficient level of QoS and IA for a given application context.

Resource allocation metrics directly influences the assessment of availability and agility, and indirectly influences timeliness (lesser amounts of resources such as bandwidth mean more time to complete a task).

Example metrics include:

- Counts of how many processer cores are unused.
- Amount of unused bandwidth over specific communication links.

Metrics in this class are often best measured on the server side of an information system. Due to the hardware/software layer abstraction implicit in system design, the user should have no visibility into the resource allocation slack other than related indirect symptoms exhibited through the application metrics.

Replication Level: Level of redundancy clearly is linked to assurance— if there are redundant ways of achieving a goal, the stakeholder is more assured that he can successfully complete his mission tasks. In that sense metrics that reflect on the redundancy present in a spatial scope (i.e., the hosts, networks, services) directly influences the availability assessments that involve that spatial scope.

Examples of metrics in this class include:

- The replication level of services running on different compute nodes. Note that a survivable system can employ various replication schemes (active, fail over etc). Usually these can be calibrated as the number of tolerated failures.

- The (edge or node disjoint) paths to services.

This metric could be measured directly by enumerating the number of replicas or distinct paths. If a direct enumeration is not possible, a statistical estimate based on replica failure rate, elapsed time and initial number of replicas can be used.

Encryption Level: The level of encryption used in messages between communicating entities directly relates to Confidentiality. Examples of this metric include:

- Whether or what percent of messages (of interest to the stakeholder) is encrypted
- Length of encryption key

Deviation from factory settings (Services/Defenses): Alterations to information systems' intended configuration (such as disabled or changed protection settings) influences the confidence stakeholders have about the system's security. If the system is intended to offer a level of security, the system, by design, must include appropriate defense mechanisms that are supposed to operate at a designated configuration. The intended configuration can be disturbed by installation of new mechanisms and configuration changes resulting from deliberate stakeholder manipulation or as results of attacker activity. DEF STAT and RES STAT metrics, focusing on deviation from "factory settings" enable consideration of this aspect in our assessment.

These metrics are associated with specific mechanisms and services. Depending on the role played by the mechanisms and spatial scope affiliation of the services, such an attribute can play a role in the assessment of any attribute.

Examples of metrics in this class include:

- A count of how much additional software has been installed on a compute node by a user since the mission started (we assume that the mission started with the "factory setting").
    - It is debatable whether all new software installation is "bad"`-- a stakeholder may install a software tool for added security. However, from the risk analysis point of view, we treat installation of new software by non-system administrator stakeholders as negative.
- The current configuration setting of defense mechanisms and services.
    - For some mechanisms it is simple—they have built in configuration levels that offer different but ordered levels of security. For example, the key length in an encryption mechanism. However, for other mechanisms, such a firewall, additional analysis will be needed to determine whether the changed configuration is weaker or stronger. Detection that the firewall rules have changed, or counting the number of rules changed is not enough, the analysis needs to determine whether additional information flow is allowed or not.

The values of these metrics are used to determine the base assessment level.

Location and Accessibility: Location of the Information Management System (IMS) within the enterprise or tactical network and the accessibility of the network itself can increase the IA risk and impact service delivery. A simple motivating example can be wireless access: adding wireless access to an enterprise improves the availability of an information service hosted in the enterprise for service consumers—they do not necessarily need to be physically on the enterprise network. However, the wireless access point may introduce a weak link in the security of the enterprise—for instance, it may use WEP (wireless encryption protocol) instead of WPA (Wi Fi Protected Access), and WEP is known to be weaker than WPA.

Example metrics in this class include:

- Whether the information system, service or network is accessible within a physical location using wired or wireless connection
    - Note, whether WEP/WPA is measured as defense presence or not
- For systems and services that are accessible remotely, the nature of the network access---is it accessible through the public Internet, military network etc.
    - Note, whether there is any IPSec or TLS is used is measured as defense presence or not
- The centrality (hop/subnet distance) of the information system within the enterprise—i.e., whether the network accessible information system is behind the DMZ or not, how many levels of indirections exist between the access point and the service

B.2 Metrics on Inherent System Properties

In this section we list metrics that measure or observe inherent system properties. Metrics of this nature do not change at runtime. These metrics are not measured at runtime, but are generally present in the system by design or implementation or because of the environment in which it is operating. The system property metrics belong to the two metric classes in our taxonomy:

- Process and Organizational Maturity
- Architecture and Infrastructure Quality

Process and Organizational Maturity (POM): This metric class considers the maturity of the software and security engineering process, and the cultural and operational practices of the organization. Based on our survey of best practices in security evaluation, we found process and organizational factors are considered in quite a large number of current security evaluation methodologies such as NSA INFOSEC Assurance Capability Model (2004), CERT/CC Security Capability Model (2005), B-Secure Security Maturity Model (2003), NIST SP 800 (Security Self Assessment Guide for IT Systems (2001). It has been observed that there are differences in the way a mature organization or a system that was a product of a mature software process handles a crisis situation compared to a less mature organization or a software system that is a product of a less matured process. The POM metrics aim to capture this aspect, but use of POM metrics is not mandatory in QIAAMU assessment framework.

An example of a POM metric is CMMI level (such as CMMI level 5), or CC/EAL certification level. For a runtime assessment approach like ours, POM metrics offer a damping factor against external events pressuring a downward revision of delivered IA. The intuition behind this argument is that the threat posed by a newly discovered vulnerability to a system at EAL7 should be lesser than what it would have been if the system was at EAL 2. Other examples include the level of the user's experience with the information system, the level of training or the clearance level required to use the system. The intuition behind these metrics is inexperienced users may incorrectly set their file permissions, improperly configure keys and/or inadvertently expose the information system to attack.

Now, what constitutes the appropriate measure of POM is heavily context dependent. Furthermore, we currently do not have any basis to determine the strength of the damping effect for each measure. Work such as the one reported in (http://www.cert.org/resilience/rmm.html) describes techniques and models to measure maturity—but does not specify what it means to stakeholders' confidence. To validate whether POM metrics influence IA and to calibrate the degree of influence, empirical studies with longer horizon is needed—this is beyond the scope of the QIAAMU work. For this reason, beyond the following generic guidelines, our framework leaves it up to the assurance engineer and the stakeholders to define and use POM metrics:

- If a POM measurement is available (e.g., CERT resilience maturity model, EAL level, or it is known that 100% of the system nodes are patched up to date) then a basic assessment based on DEF STAT values can be incremented by a constant fraction (i.e., if you are in level *l*, this boost does not get you to level (*l+1*)). For example, let us assume that the possible values of relevant DEF STAT system conditions for a given assessment may lead to *N* configurations, which is then organized into *m* levels (i.e., 0,1,…m-1) where *m* levels sufficient to cover the stakeholder's mission requirements. Let us also assume that the current values of the system conditions are such that the assessed value for a given attribute over a given scope is 1. Based on a high RMM rating (a POM measurement), this value can be boosted to be 1.5. This has utility only in the cases where two systems with similar survivability architecture and defense mechanism configurations are compared. For continuous runtime assessment, this constant increment to the base assessment has no impact, because it is added to the base assessment always.
    - Use of POM metrics in the above manner is not selective to IA attributes. Unlike the way RES STAT influences the assessment of A, POM revises the assessment of C, I, A, and A/A.
- If a POM measurement is available, it can be used to dampen the impact of EXT and DEF REP system condition values. For example, recall that EXT and DEF REP metrics are used for assessment of C, I and A/A. If an EXT or a DEF REP value was to depress the base assessed value by factor *F,* a high POM measurement would mean that the depression be .5F. This mode is useful for continuous runtime assessment.

Architecture and Infrastructure Quality (AIQ): We have shown in previous work (see [44], [45] in the bibliography of this Final Report) that the resiliency of a system against malicious attacks is strongly related to the system's survivability architecture. We aim to capture the influence a well architected survivable system has on the stakeholders' confidence in the system. This metric class was designed to support that goal. An example of this metric is Common Weakness Enumeration (CWE). CWE highlights the connection between software quality and security assurance. If a system has lesser CWEnumber than another, we claim that the former instills more confidence in the stakeholders, i.e., delivers more assurance.

As with the POM metrics, no formal basis exists to calibrate the influence. The same guidelines for using POM are applicable to the use of AIQ metrics as well.

B.3 Metrics on Operational and Inorganic System Aspects

In this section we list measurable aspects that are driven by factors external to the information system, but part of the system's operational context. These metrics cover:

- Performance of defense mechanisms
    - What they are experiencing: this is DEF REP, and influencing IA attributes and timeliness
    - How effective they have been: this is DEF EFF, and influencing the IA attributes only
- Load: affecting timeliness and availability
- Human factors:

    - Adversary capability and value : this is EXT, affecting IA attributes
    - User (stakeholder) experience: this is POM and influencing IA attributes

Recent Experiences (of Defense and Services): The recent experience/health or status information of the defenses and services provide valuable clues about the overall assurance of the system. The metrics in this category considers the reports (logs, alerts) issued by both the defenses and services in order to extract relevant information that can influence our assessment. The extracted information is distinct from and in addition to the ON or OFF state/configuration of the defense mechanisms and services. Examples include:

- Definitive alert counts (e.g., matching signatures) organized in terms of the affected spatial scope and security attribute (e.g., a sensor declaring ping of death attack targeted at specific IP and port).
    - How this metric is used in the assessment depends on whether the sensor is a firewall, which does intrusion prevention (i.e., block) in addition to intrusion detection or just a detector. The latter case would see a downward revision of the assessment. The former case is similar to the policy violation case discussed next.
- Policy violation attempts issued by policy enforcement mechanisms organized by the security attribute and spatial scope the policy is trying to protect. We assume that a background level of violation attempts as normal, and when a significant surge from that background level is observed, we revise the assessment of the affected spatial scope and security attribute downward
- Uncertain alerts from defense mechanisms/event logs from services: Most of the alerts issued by IDSs today are not definitive. Although signature based IDSs are more reliable they do have false positive and false negatives, especially when attackers actively attempt to change or hide their signatures (polymorphic attacks). Anomaly-based IDSs are inherently prone to false alerts. In many cases the IDSs and a correlation mechanism feeding off multiple IDSs attempt to aggregate and rank the raw low-level alerts to something that is more useful for our analysis. For example, output from a correlator may indicate a target (spatial scope), affected security attribute, severity and confidence value. The high severity and high confidence alerts should see a downward revision of the assessment of the affected spatial scopes.
    - We note that the normalization of the reports is a critical problem-- IDSs or correlators cannot be expected to use a common standard.

The thresholds, the extent of downward revisions, and normalization of severity/confidence need to be calibrated in a context specific and empirical way by the assurance engineer.

Defense Effectiveness: The intuition behind the DEF EFF metrics is that stakeholders should be less assured about the system when it is known that the defense mechanisms are attempting to respond to a situation and yet the situation persists. In our prior work, we have demonstrated that by watching alerts and logs of defense mechanisms it is possible to determine whether defense is being successful. In case of automated response, we expect the mechanism mounting the response to include the condition or state that it is responding to or trying to rectify, which can then be monitored to determine whether the automated response is being effective or not. In case of an operator initiated response, we rely on the operators to indicate whether their responses are being effective by setting system condition value.

At the lowest level, defense effectiveness can be measured per "response". Dedicated log watchers will start a thread when a mechanism mounts a response, extracts the information about the condition the response is trying to remedy and try to monitor the system for that condition. This is not a simple task. It is much easier to determine whether a host or a reachability is up or down, but it is not feasible to determine whether conditions like a host or a service is corrupt. Even for simpler conditions like a host or service down, there is no guarantee that this information will be available in the local or peer blackboards, or even any of the ESM databases. Therefore, we rely on the fact that the defense mechanisms will mount additional responses if the initial response is ineffective. We further assume that a "subsumes" relation can be imposed on the perceived system state represented by our assessment values or logs captured by existing sensors (i.e., the current condition is same, worse or better than the previous). Given this, the log monitors will be able to ascertain whether defenses are effective.

Our initial plan is to use the DEF EFF metric at a gross level, i.e., a system condition indicating whether the overall defense is being effective. There are at least two implementation possibilities. In one, the local log monitoring processes at each defense mechanism report directly to a global DEF EFF system condition, which employ context specific rules to determine overall effectiveness, e.g., if >70% mechanisms are reporting effective, then the overall defense is effective. In another, log monitors local to each defense mechanism will have a dedicated DEF EFF system condition, and a global DEF EFF system condition will use the values of these system conditions to determine the overall defense effectiveness.

Load: The load on the system, specifically congestion and heavy usage of system resources directly impacts the availability attribute. Indirectly, load may influence how defense mechanisms behave—and thereby influence the IA or QoS attributes that these mechanisms serve. Examples of metrics from this class include:

- The number of users logged in.
- The volume of traffic (in GB per sec) being handled by a server (e.g., ftp, or image server)

These measurements are readily available from ESMs, and are treated as RES STAT metrics in our assessment. Initially, we have restricted the use of RES STAT on the assessment of availability only.

Adversary Capability and Value: A potent adversary has a greater potential impact, and if the information presents a valuable potential target to intelligent adversaries, then the potential for adversary action is greater. Similarly, if a new vulnerability or exploit is discovered, and the system is susceptible to that, the potential of adversary activity rises. In many cases, the adversary wants to hide behind other ongoing attacks such as botnet activities or Internet worms. Recent bots or worms are known to initiate multiple virus activities fully expecting that they will be detected and possibly stopped just to create a diversion to hide the real malicious interactions. Presence and level of factors such as virus activities or disclosure of new vulnerabilities therefore should lead to a downward revision of the assessed values. Measuring this metric relies on observations of relevant events that happen in the environment in which the system operates.

Example measurements and observations include:

- Amount, scope and severity of CERT advisories and vendor-issued vulnerability announcements: large number of high severity vulnerabilities with sweeping scope increases the potency of all kinds of adversaries.

- DEFCON or DHS threat level type indications that may imply increased risk to DoD information systems. These threat levels often aggregate the raw information from CERT or vendor furnished advisories, i.e., a raised threat level may be the result of "large number of high severity vulnerabilities with sweeping scope" situation mentioned in the previous bullet.

These measurements and observations are treated as EXT metrics. EXT metrics impacts the base level assessed values of C, I and A/A negatively.

Use Experience Level: User's level of training and experience impact how assured and risky the system and its use in the mission remains when the system is under attack. Intuitively, a well trained and seasoned operator is likely to make more reliable choices and not likely to make errors that an inexperienced operator will make. However, how to measure human expertise and behavior under stress is a different topic (psychology?), and we are not focusing on how to use that measurement in assessing the system's runtime assurance in our current work.

APPENDIX C – QIAAMU DELIVERABLES

This section enumerates the documented deliveries that we made under the QIAAMU contract, by category and in chronological order.

C.1 Technical Reports

1. Literature Study
2. Taxonomy and methodology
3. Interim test and evaluation
4. Interim Final Report
5. System architecture and design document.
6. Test and evaluation document

C.2 Software Prototypes and Documentation

1. Installation and User Guide Documentation
2. QIAAMU Software source and executable

C.3 Presentation Material

1. Kickoff, site visits, other meetings

C.4 Monthly Status Reports

1. Various throughout the period of performance.

APPENDIX D – QIAAMU PUBLICATIONS AND PRESENTATIONS

1. Partha Pal and Patrick Hurley. Assessing and Managing Quality of Information Assurance. NATO IST Symposium on Cyber Security and Information Assurance. April 26-27 2010, Tekirova-Antalya, Turkey. (Conference rescheduled to Estonia due to Icelandic volcanic activity)
2. Partha Pal, Rick Schantz, and Joseph Loyall, Middleware for Runtime Assessment of Information Assurance, The ACM/IFIP/USENIX 11th International Middleware Conference (Middleware 2010), Industry Track, Bangalore, India, Nov 29- December 3, 2010
3. Amelia Fedyk, Michael Atighetchi, Partha Pal Leveraging ESM Platforms for Continuous Mission-Oriented Assessment of Information Assurance. The 6th International Conference on Network and Services Management, Niagra Falls, Canada, October 2010
4. Partha Pal, Rick Schantz, Michael Atighetchi, Kurt Rohloff, Nathan Dautenhahn, William Sanders. Fighting Through Cyber Attacks: An Informed Perspective toward the Future. Workshop on Survivability in Cyber Space (Invited paper). April 12 2010, Stockholm, Sweeden.
5. Partha Pal, Kurt Rohloff, Michael Atighetchi, Rick Schantz. Managed Mission Assurance: Concept, Methodology and Runtime Support. Workshop on Mission Assurance: Tools, Techniques, and Methodologies at the 2nd IEEE International Conference on Privacy, Security, Risk, and Trust. August 20-22 2010, Minneapolis, Minnesota, USA.
6. Patrick Hurley, Partha Pal, Mathew Tan Creti, Amy Fedyk. The 5th Worksop on Recent Advances in Intrusion Tolerant Systems at the 41st IEEE/IFIP International Conference on Dependable Systems and Network (DSN), June 27, 2011, Hong Kong, China.
7. Hala Mostafa, Partha Pal, Patrick Hurley. Message Passing for Distributed QoS-Security Tradeoffs. Fifth International Workshop on Optimisation in Multi-Agent System (OptMAS) at AAMAS 2012, Valencia, Spain, June 2012.
8. J. Zhang, R. Berthier, M. Bailey, P. Pal, W. H. Sanders, and F. Jahanian, "Safeguarding Academic Accounts and Resources with the University Credential Abuse Auditing System," in Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2012), Boston, Massachusetts, USA, June 25-28, 2012
9. J. Zhang, R. Berthier, M. Bailey, W. Rhee, W. H. Sanders, P. Pal, and F. Jahanian. (12ZHA02) Proceedings of the 5th Workshop on Cyber Security Experimentation and Test (CSET'12), Bellevue, Washington, Aug. 6, 2012
10. Hala Mostafa, Partha Pal and Patrick Hurley. Message Passing for Distributed QoS-Security Tradeoffs, Special issue of the Computer Journal, British Computer Society, in review/update.
11. Hala Mostafa, Partha Pal, Nate Soul, Nick Hoff, Emily Hahn and Patrick Hurley. Applying Distributed Optimization for QoS-Security Tradeoff in a Distributed Information System. AAMAS 2013 (under review).

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

API	Application Programming Interface
APS	Advanced Protected Services
AFRL	Air Force Research Laboratory
AS	Application Server
ATD	Advanced Technology Demonstration
BBN	Bolt, Beranek, and Newman; now Raytheon BBN Technologies, a company with headquarters in Cambridge, Massachusetts
C, I, A	Confidentiality, Integrity, Availability
CPU	Central Processing Unit
CZ	Crumple Zone
DARPA	Defense Advanced Research Projects Agency
DDOS	Distributed Denial of Service
DoD	Department of Defense
DOS	Denial of Service
DPASA	Designing Protection and Adaptation into Survivability Architectures
FW	Firewall
GB	Gigabyte
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
IP	Internet Protocol
IPSec	Internet Protocol Security
IT	Information Technology
JDBC	Java Database Connectivity
JDK	Java Development Kit
JMS	Java Message Service
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
JVM	Java Virtual Machine
LAN	Local Area Network
LTE	Limited Technology Experiment
MAPE	Monitor-Analysis-Planning-Execute
NIC	Network Interface Card
OS	Operating System
OSSEC	Open Source Security
R&D	Research and Development
RAM	Random Access Memory
SBT	Security Benefit Target
SE Linux	Security Enabled Linux
SOA	Service Oriented Architecture
SPA	Single Packet Authorization
SPOF	Single Point of Failure
SQL	Structured Query Language

SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TRL	Technology Readiness Level
VM	Virtual Machine
VPN	Virtual Private Network
WAN	Wide Area Network
WS	Web Services
WSDL	Web Services Description Language
XML	Extensible Markup Language